

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A COMPARISON OF ANALYSIS IN DIS AND HLA

by

Steven D. Knight

June 1998

Thesis Advisor:

Arnold H. Buss

Second Reader:

William S. Murphy, Jr.

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>			
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE A COMPARISON OF ANALYSIS IN DIS AND HLA		5. FUNDING NUMBERS	
6. AUTHOR(S) Steven D. Knight			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) TRADOC Analysis Center - Monterey Monterey, CA 93943-5101		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>As the Department of Defense (DoD) continually relies more on Modeling and Simulation (M & S) for testing, analyzing, and training, issues of interoperability have become one of the most important concerns. As such, DoD adopted the Distributed Interactive Simulation (DIS) protocol in 1991. Although successful in many aspects, DIS is limited by available information from models, memory and network requirements, and analytical tools available. Therefore, in 1996 the Defense Modeling and Simulation Office (DMSO) released the High Level Architecture (HLA), an object-oriented approach to interoperability.</p> <p>This thesis compares these different approaches to analysis to determine functionality in terms of gathering, processing, and reporting on analytical questions in both environments. To compare DIS and HLA analysis, three simulation runs were conducted: Janus vs. Janus in DIS, HLA without an Analysis Federate, and HLA with an Analysis Federate. The Analysis Federate is an HLA-compliant software package that gathers and processes information for analysis requirements. The results of the three simulation runs and subsequent analysis demonstrated the techniques and approaches for each infrastructure. The resulting comparison between them show HLA with the Analysis Federate is the easiest and most functional tool.</p> <p>The Analysis Federate fills an analysis void currently in HLA and by implementing it with the study question model tree methodology, an analyst will be more effective and be able to provide real-time feedback.</p>			
14. SUBJECT TERMS Analysis Federate, Distributed Interactive Simulation (DIS), High Level Architecture (HLA), Federation Object Model (FOM), Simulation Object Model (SOM), Janus, Gateway, Study Question Model Tree, Federation, Run-Time Infrastructure (RTI), Protocol Data Unit (PDU), PDU Adapter Software System		15. NUMBER OF PAGES 97	
		16. PRICE CODE	

(PASS)

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500
89)

Standard Form 298 (Rev. 2-

Prescribed by ANSI Std. Z39-18
298-102

Approved for public release; distribution is unlimited.

A COMPARISON OF ANALYSIS IN DIS AND HLA

Steven D. Knight
Captain, United States Army
B.S., United States Military Academy, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
June 1998**

Author:

Steven D. Knight

Approved by:

Arnold H. Buss, Thesis Advisor

William S. Murphy, Second Reader

Richard E. Rosenthal, Chairman
Department of Operations Research

ABSTRACT

As the Department of Defense (DoD) continually relies more on Modeling and Simulation (M & S) for testing, analyzing, and training, issues of interoperability have become one of the most important concerns. As such, DoD adopted the Distributed Interactive Simulation (DIS) protocol in 1991. Although successful in many aspects, DIS is limited by available information from models, memory and network requirements, and analytical tools available. Therefore, in 1996 the Defense Modeling and Simulation Office (DMSO) released the High Level Architecture (HLA), an object-oriented approach to interoperability.

This thesis compares these different approaches to analysis to determine functionality in terms of gathering, processing, and reporting on analytical questions in both environments. To compare DIS and HLA analysis, three simulation runs were conducted: Janus vs. Janus in DIS, HLA without an Analysis Federate, and HLA with an Analysis Federate. The Analysis Federate is an HLA-compliant software package that gathers and processes information for analysis requirements. The results of the three simulation runs and subsequent analysis demonstrated the techniques and approaches for each infrastructure. The resulting comparison between them show HLA with the Analysis Federate is the easiest and most functional tool.

The Analysis Federate fills an analysis void currently in HLA and by implementing it with the study question model tree methodology, an analyst will be more effective and be able to provide real-time feedback.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION.....	1
A. BACKGROUND	1
1. General	1
<i>a. Modeling and Simulation</i>	<i>1</i>
<i>b. History (Stand-alone to Distributed Simulations).....</i>	<i>3</i>
2. Distributed Interactive Simulation (DIS).....	5
3. High Level Architecture (HLA)	6
B. JANUS COMBAT SIMULATION MODEL	8
1. Stand-alone	9
2. PDU Adapter Software System (PASS)	10
C. PROBLEM DESCRIPTION.....	12
1. Analysis Tools	12
<i>a. Post Processors.....</i>	<i>12</i>
<i>b. Data Loggers (DIS)</i>	<i>13</i>
<i>c. Analysis Federate (HLA).....</i>	<i>14</i>
2. Advantages and Limitations of the Analysis Tools Available	15
3. Analytical Approaches in DIS and HLA.....	17
D. THESIS STATEMENT AND OUTLINE	17
II. METHODOLOGY.....	19
A. COMPARISON OF ANALYTICAL APPROACHES	19
1. Janus vs. Janus in DIS.....	19
2. Janus vs. Janus in HLA without an Analysis Federate	21
3. Janus vs. Janus in HLA with an Analysis Federate	21
B. SCENARIO DEVELOPMENT	22

C. ANALYSIS QUESTION DEVELOPMENT	23
D. SOFTWARE AND HARDWARE REQUIREMENTS	25
1. Janus as the Base Model.....	25
2. DIS Setup	26
3. HLA Setup.....	27
a. HLA Gateway.....	30
b. Analysis Federate	30
III. RESULTS	33
A. SCENARIO DEVELOPMENT	33
1. Exercise Scenario.....	33
a. General Situation	34
b. Specific Situation.....	34
c. Current Situation	35
2. General	36
B. ANALYSIS QUESTIONS	40
C. COMPARATIVE APPROACHES	46
1. Janus vs. Janus in DIS.....	46
a. Data Collection	46
b. Results	49
2. Janus vs. Janus in HLA without an Analysis Federate	54
a. Data Collection	55
b. Results	56
3. Janus vs. Janus in HLA with an Analysis Federate	58
a. Data Collection	59
b. Results	63
D. DISCUSSION.....	65
IV. CONCLUSIONS AND RECOMMENDATIONS	70
A. CONCLUSIONS	70

B. RECOMMENDATIONS.....	72
C. FUTURE WORK.....	72
APPENDIX A. PROTOCOL DATA UNIT (PDU) FORMAT.....	74
APPENDIX B. JANUS DATABASE	78
APPENDIX C. JANUS POST PROCESSOR REPORTS	80
APPENDIX D. HLA GATEWAY MODIFICATIONS	84
APPENDIX E. SCENARIO SIDE ONE AND TWO BREAKDOWN	86
APPENDIX F. PERL CODE FOR KILLER/VICTIM DATA	88
APPENDIX G. PERL CODE FOR ROUND TYPE DATA.....	92
APPENDIX H. GLOSSARY OF ACRONYMS.....	94
LIST OF REFERENCES.....	96
INITIAL DISTRIBUTION LIST	98

LIST OF FIGURES

FIGURE 1: PASS IN A DIS ENVIRONMENT [9]	11
FIGURE 2: STUDY QUESTION MODEL TREE.	25
FIGURE 3: DIS SETUP.....	27
FIGURE 4: HLA SETUP.	29
FIGURE 5: INITIAL U.S. FORCE DISPOSITION.	38
FIGURE 6: INITIAL SERBIAN FORCE DISPOSITION.	39
FIGURE 7: STUDY QUESTIONS.....	41
FIGURE 8: UNIT STRENGTH MOES AND DATA COLLECTION REQUIREMENTS.....	42
FIGURE 9: UNIT’S LER.....	43
FIGURE 10: CLASS V (AMMUNITION) STATUS.	44
FIGURE 11: CLASS III (FUEL) STATUS.....	45
FIGURE 12: JANUS VS. JANUS IN HLA WITHOUT AN ANALYSIS FEDERATE.....	54
FIGURE 13: JANUS VS. JANUS IN HLA WITH AN ANALYSIS FEDERATE.	59
FIGURE 14: OPERATIONAL VIEW WITH FOCUS SETS.....	61
FIGURE 15: PRIMARY LIST FOR A FOCUS SET.....	62
FIGURE 16: JANUS DATABASE HIERARCHICAL DIAGRAM. [15]	78

LIST OF TABLES

TABLE 1: KILLER/VICTIM SCOREBOARD.	49
TABLE 2: ROUND COUNT BY ENTITY.	50
TABLE 3: PERCENT REMAINING FOR EACH PLATOON IN A COMPANY.	51
TABLE 4: A COMPANY’S LER.	52
TABLE 5: AMMUNITION EXPENDITURE.....	53
TABLE 6: A COMPANY’S STRENGTH.	56
TABLE 7: A COMPANY’S LER.	56
TABLE 8: A COMPANY’S AMMUNITION EXPENDITURE.....	58
TABLE 9: A COMPANY’S STRENGTH REMAINING.....	64
TABLE 10: A COMPANY’S LER.	64
TABLE 11: A COMPANY’S AMMUNITION EXPENDITURE.....	65
TABLE 12: STATISTICS FOR OVERALL DATABASE. [15]	79

EXECUTIVE SUMMARY

As the Department of Defense (DoD) continually relies more on Modeling and Simulation (M & S) for testing, analysis, and training, issues of interoperability have become one of the most important concerns. Beginning with the success of the Simulation Networking (SIMNET) program in 1984, in which models interacted in a distributed environment, DoD has continually incorporated emerging technologies to improve interoperability in a distributed environment.

The first DoD-wide standard was the Distributed Interactive Simulation (DIS) protocol adopted in 1991. Under DIS, models broadcast Protocol Data Units (PDUs) over an area network and received PDUs from other models. The PDUs attempted to contain sufficient information to allow various models the ability to represent entities and events within each model. Although successful in many aspects, DIS is limited by available information from models, memory and network requirements, and analytical tools available. Therefore, in 1996 the Defense Modeling and Simulation Office (DMSO) released the High Level Architecture (HLA), an object-oriented approach to interoperability. HLA requires models, or federates, to publish and subscribe to objects, interactions, attributes, and parameters specified in the Federation Object Model (FOM).

Analysis is conducted differently in DIS and HLA due to the differences between broadcasting and publish/subscription requirements. This thesis compares these different approaches to analysis to determine functionality in terms of gathering, processing, and reporting on analytical questions in both environments.

The PDUs broadcast in DIS are sent over a User Datagram Protocol/Internet Protocol (UDP/IP) network. This means any model listening to the right port can receive

all of the PDUs, and hence exercise information. By recording the PDUs with the use of a data logger, information can be stored and data collection requirements extracted after the exercise is complete. Therefore, analysis questions are typically not developed or data requirements determined until after the exercises.

HLA uses a different approach than DIS. In HLA, a model registering with the federation publishes and subscribes to required information. No analytical tools are inherent in HLA, currently leaving only the individual models post processor reports. However, an Analysis Federate has been proposed to perform analysis under HLA. This federate would subscribe to information required to answer analysis questions and measures of effectiveness (MOEs). Additionally, this required information must be known prior to registering, meaning that the analysis questions must also be developed prior to registering.

A proposed question development process is called the study question model tree and was used in this thesis. The study question model tree begins with the overall objective of the exercise and works down through study questions and MOEs until the required data is determined. Then the subscription requirements can be determined from the required data. Once the data is gathered during the exercise, the process reverses until all the study questions have been answered and the objective met.

To compare DIS and HLA analysis, three simulation runs were conducted: Janus vs. Janus in DIS, HLA without an Analysis Federate, and HLA with an Analysis Federate. Each simulation run used the same scenario and analysis questions. The scenario was based on Bosnia and incorporated the factors of realism, flexibility, and

tactical soundness. The analysis requirements were developed using the study question model tree methodology and used for all three simulation runs.

The results of the three simulation runs and subsequent analysis demonstrated the techniques and approaches for each infrastructure. The resulting comparison between them show HLA with the Analysis Federate is the easiest and most functional tool. It provides a workstation that an analyst can learn to use in a short amount of time and still present quality results. It also provides the opportunity for real-time analysis. This is a big advantage over the other techniques since feedback can be provided to the commanders while the exercise is still executing.

The overall recommendations from this study are twofold. First of all, incorporate the Analysis Federate into all HLA federation requiring analysis. The Analysis Federate developed by the TRADOC Analysis Center (TRAC) – Monterey provides the added functionality of interoperability within any federation. The second recommendation is incorporate the study question model tree methodology to approaching analysis, resulting in a more proactive analyst.

As DoD continues to progress towards HLA, further study on time latency issues, data processing in the Run-Time Infrastructure (RTI), and standardized reports in the Analysis Federate deserve consideration. Each of these areas impact on the overall results of the simulation run by either increasing the accuracy or reducing the amount of processing and calculations that would otherwise be necessary external to the Analysis Federate.

In summary, the Analysis Federate fills an analysis void currently in HLA. By implementing it with the study question methodology, an analyst will be more effective and be able to provided real-time feedback.

I. INTRODUCTION

A. BACKGROUND

1. General

Recent advances in computer technology and increased use of the Internet have revolutionized the manner in which commercial industry and the Department of Defense (DoD) do business. With reduced budgets and increased mission requirements, DoD is looking for ways to minimize costs while maintaining readiness. This is no small task, since the potential areas of conflict are not as concrete as they were before the end of the Cold War. Potential future U.S. missions range from disaster relief to peace keeping to all out war. Each of these potential missions requires specialized training, equipment, and other supplies and support. It is simply too costly to meet all these needs within the current budget constraints.

a. Modeling and Simulation

Modeling and Simulation (M & S) has helped bridge the gap between the budget and maintaining readiness. M & S consists of techniques and tools for testing, analyzing, or training in which real-world and conceptual systems are reproduced by a model. [8] M & S allows units and agencies to test various tactics, techniques, and procedures (TTP) or equipment without actually deploying or firing expensive weapons. Additionally, M & S can significantly reduce the acquisition time and cost of new weapon systems and platforms. The amount of time and money this process can save in both the short and long term depends on the scale of the implementation.

The DoD uses three kinds of simulations: live, constructive, and virtual. First, live simulations involve real people using real systems in a synthetic environment

short of a conflict. Examples of this include a rotation at the National Training Center, an emergency room triage exercise, and a live-fire attack in urban terrain. Next, virtual simulations involve real people using simulated systems in a synthetic environment. For example, the Battle of 73 Easting and M1 Abrams tank simulations are considered virtual simulations. Finally, constructive simulations are simulated people and systems in a synthetic environment. They usually rely on computer-based models implementing algorithms and mathematical models. [2]

Constructive simulations can be broken down further into high and low resolution models. Although there is not a clearly defined line distinguishing one from the other, high resolution models typically define units and entities at battalion level or below. Low resolution models, on the other hand, aggregate units and combat potential at brigade level or above. The analysis requirements, time, and software/hardware requirements determine which constructive model to use. M & S developers create these kinds of simulations to represent actual combat scenarios and entities as realistically as possible.

Two DoD-mandated infrastructures, Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) attempt to seamlessly link the various kinds of simulations together. Ideally, a unit training in the field could interact with a constructive simulation representing forces on their flanks and rear and with a virtual simulation of a platoon training on M1 Abrams tank simulators. DIS and HLA's goal is to increase interoperability between the different simulations, regardless of type.

b. History (Stand-alone to Distributed Simulations)

Combat models have been around for a long time. The earliest known model is credited to Sun Tzu about 5000 years ago. His model was a wargame known as Wei Hai that allowed contestants to maneuver their armies of colored stones around on a playing surface. [3] More recently, commanders in World War II used large board games to replicate battles in order to determine their best course of action. Since then, computer technology has developed to the extent that both small and large-scale combat scenarios can be fought in both high and low resolution without any actual rounds being fired.

The original computer models were built as stand-alone simulations. Each model was self-contained and would fight its scenario without interacting with any other computers. It was not until the 1980's that a distributed environment became possible:

It was not until the success of the Simulation Networking (SIMNET) program that it was considered technically feasible and economically affordable to use virtual prototypes in a much broader sense for both training and acquisition. SIMNET demonstrated that core technologies were mature enough to support large scale, interactive, real time networks of manned simulators, emulators, and automated forces. [2]

SIMNET began in 1984 and proved that the technology existed to allow several models to interact. No longer did one large stand-alone model need to be built for each training or analysis requirement. Instead, smaller, specialized models could be developed that could leave irrelevant interactions to other models and concentrate on their own specific training or analysis requirements. In this way, models could be reused in different simulations with only minor changes to initial parameters.

As M & S has moved into a distributed environment, interoperability has become one of the major concerns of M & S developers and users. To increase interoperability, DoD sought to develop a common framework to provide a standard set of operating procedures for model interaction. These operating standards addressed issues ranging from passing information on objects, attributes, parameters, and interactions to hardware requirements. The frameworks were designed to be strict enough to allow the passage of information quickly, accurately, and efficiently while still allowing enough flexibility so both legacy and newly developed models could accomplish their purpose in a distributed environment.

In 1991 the Army Science Board Study on Simulation Strategy recommended adopting DIS. Since then, DIS oversight, functional management, and technical management have been established. DIS has been developed and refined since then using existing and emerging technologies. The idea of DIS was to broadcast and receive information between M & S over a local or wide area network in order to simulate various scenarios.

In 1996, DoD reassessed the issue of interoperability based on current technology. As a result, DoD is changing distributed infrastructures from DIS to HLA. As with DIS, HLA attempts to take advantage of current and emerging technologies. Although still in its infancy, HLA implementation is rapidly progressing towards the mandate that all M & S be HLA compliant by 2001 for all DoD simulations, issued by Dr. Paul Kaminski, Under Secretary of Defense for Acquisition and Technology in DoD Directive 5000.59. [3]

2. Distributed Interactive Simulation (DIS)

After the success of SIMNET, DIS was developed to broaden the concept of model interoperability. “The essence of DIS is the creation of a synthetic environment within which humans and simulations interact at multiple networked sites using compliant architecture, modeling, protocols, standards and data bases.” [2] Simulations communicate with each other using DIS protocol by broadcasting protocol data units (PDUs) over a computer network. “The purpose of the PDU is to facilitate the electronic transfer of data between simulations with different software.” [5] The PDU formats are specified in the DIS protocol standards. Each PDU has a header that identifies the exact exercise, machine, and time of a particular event occurrence. The body of the PDU contains information about the event that occurred. A PDU can represent events ranging from firing to detonation to electromagnetic emission and contains sufficient information so that other models can replicate the event.

To run a DIS exercise, models distributed over a local or wide area network (LAN or WAN) send PDUs using the User Datagram Protocol/Internet Protocol (UDP/IP). A Model broadcasts PDUs for one of two reasons: an event occurrence or “heartbeat” rate updates. Events include an entity movement of a pre-specified distance or the firing of a weapon. A PDU is broadcast whenever there is a significant change in an entity’s state. A “heartbeat” rate update occurs at a constant rate determined prior to the execution of the simulation run, typically every 4 or 5 seconds. Heartbeat updates ensure that all models in the exercise have the same simulated time for each event.

The DIS infrastructure does not explicitly specify how to conduct analysis or provide any analytical tools. However, since PDUs are broadcast over the network, data

loggers can listen and record them in a logger file. Once an exercise is complete, required data can be extracted from the logger file and analyzed. Another analytical tool available to the analyst is the post processor found in the individual models. The post processor provides analysts with a predetermined list of possible reports. In neither case are analysts forced to determine the exact study questions and required data prior to the exercise. Instead, at the end of a simulation run, the analyst has the “entire” battle stored in the data logger. At this point, as long as the required information is included in the PDUs, it is just a matter of extracting what is needed. Otherwise, the study question requiring that information cannot be answered.

3. High Level Architecture (HLA)

DIS was DoD’s first military-wide attempt towards model interoperability across a distributed network. Although successful in many aspects, DIS is limited by available information from models, memory and network requirements, and analytical tools available. Therefore, to increase interoperability among simulations and further promote reuse of simulations and their components, DoD is transitioning from the DIS protocol to an architecture, HLA, still in the development and initial implementation stage. A federation under HLA is a named set of interacting federates (simulations, C4I system interfaces, data collectors, etc.) with a common federation object model (FOM), and supporting Runtime Infrastructure (RTI) software that are used as a whole to achieve a specific objective. In HLA, federates must subscribe and publish their data requirements, unlike DIS which broadcasts all its data across the network. Only subscription and published information is passed between the models in an HLA federation.

The FOM can be viewed as a contract between federates that describes the type and names of data they agree to share through the RTI during a federation execution. It consists of Object Model Templates (OMT) which specify object class structure, interactions, attributes, parameters, and data types. Each federate also has object model tables constructed similarly to the FOM called a simulation object model (SOM). The SOM outlines the same information as the FOM except only for its federate. In other words, the SOM does for the federate what the FOM does for the federation.

To run an HLA exercise, each model, or federate, first registers with the federation. During this process, the federate specifies its subscription requirements to the federation's RTI. The RTI records these requests and checks the FOM's Federation Execution Data (FED) file to ensure compliance with the federation's standards. The FED file contains the data standards for the objects, interactions, parameters, and attributes. Once the exercise begins, the RTI directs the flow of information between the models that are registered to ensure each federate gets the information it requested.

As with DIS, HLA does not have analytical tools built into its infrastructure, which presents an obstacle for HLA analysis. With HLA's publishing and subscribing requirements, a passive data logger is no longer a viable option. One proposal to overcome this deficiency calls for the development of an HLA federate designed specifically to perform analysis, an analysis federate. [6] This federate would be separate from the other models within the federation and would subscribe only to the information the analyst needs to answer his study questions. Training and Doctrine Command (TRADOC) Analysis Center (TRAC) – Monterey developed an Analysis Federate which provides the data collection functionality proposed in the referenced

paper. However, the actual implementation of the delivered Analysis Federate is significantly different from a systems design point of view. [16] This thesis uses the TRAC – Monterey implementation of the Analysis Federate. A detailed discussion of the Analysis Federate and methodology is included later in this chapter in section C. 1. c.

B. JANUS COMBAT SIMULATION MODEL

DIS and HLA are just frameworks allowing models to interact, each approaching interoperability and analysis differently. Models range from high resolution models, such as the Joint Conflict and Tactical Simulation (JCATS), to low resolution models, such as the Joint Theater Level Simulation (JTLS). The degree of resolution provides trainers and analysts different capabilities and functionality. DIS and HLA do not distinguish between which type of resolution models are being used and allow the use of both. Instead, individual models are the building blocks of the distributed environment.

Janus is a popular simulation that has been widely used since the 1970's. Originally designed as a nuclear effects simulation, Janus is now a high resolution, interactive, stochastic, ground combat simulation focusing on brigade and below sized units. [8] Janus allows analysts and commanders to test tactics, techniques, and procedures for training, contingency planning, analysis, and acquisition. In turn, this can increase the training level of leaders and reduce acquisition costs of new equipment. Janus can be run either as a stand-alone model or, using recently developed software, in a distributed mode. Additionally, the TRADOC Analysis Center in Monterey (TRAC – Monterey) is currently working towards converting Janus to HLA and modifying it so it can be used on a PC. TRADOC plans to continue using Janus in future training and analysis.

1. Stand-alone

There are four major phases in conducting analysis using Janus: building a scenario, running the scenario, using the Janus Analyst Workstation (JAWS), and analyzing post processor reports. Together these parts make Janus an effective tool for conducting training and analysis.

Janus provides a robust database for entering various parameters necessary to build a scenario. A detailed breakdown of the database is included in Appendix B. An extensive list of parameters allows the user to enter various characteristics about their specific scenario. These range from weapon capabilities to sensor types to terrain and weather characteristics. Once these parameters are entered, the user can then enter the specific tactical scenario including movement routes, minefield locations and artillery targets, to name a few. The scenario can be built to replicate the user's tactical and equipment situation for their analysis requirements.

Once the scenario is built, Janus is ready to run the scenario. It allows for human interaction during run time or for the simulation to run without a human-in-the-loop. Leaders can "issue" orders by using the command selections available during the actual run. For example, fire missions can be executed, movement routes modified, and weapon orientations changed during execution. Intelligence reports and information reports are available to the user during execution but provide only a limited amount of information, such as location and number killed by equipment type. The primary purpose of these reports is to add training value to the simulation. The battle can be saved whenever necessary to give the analyst or trainer more flexibility for their requirements.

At the termination of the simulation run, the battle can be replayed using JAWS, which allows the user to conduct after battle analysis or after action reviews. Users can replay the entire battle as it happened or just have it show specific events such as detections. [8] This can prove to be very valuable for the user in their analysis or training, especially when looking for cause or effect of specific actions on final outcomes.

The Janus post processor, perhaps the most useful to the analyst and trainer, prepares reports summarizing events that occurred during the simulation run. However, the amount of information available to the analyst is limited to a predetermined list of available reports. A more detailed discussion of the post processor reports follows in section C. 1. a. of this chapter.

2. PDU Adapter Software System (PASS)

Janus was made DIS compliant by the addition of a software program called the PDU Adapter Software System (PASS), previously called the World Modeler. Together Janus and PASS form the Distributed Interactive Simulation Constructive System (DISCS), originally termed JLINK. The following figure shows how PASS works in the DIS environment.

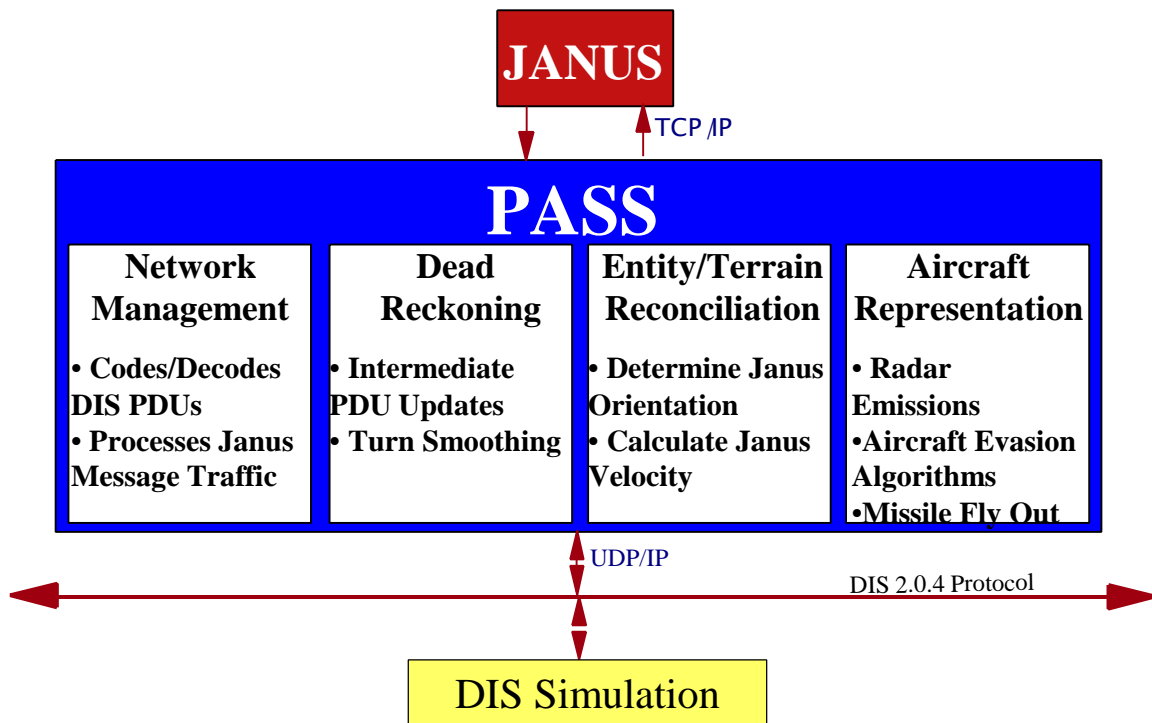


Figure 1: PASS in a DIS environment [9]

PASS translates the Janus protocols to DIS PDUs and visa versa, communicating with Janus using the Transmission Control Protocol/Internet protocol (TCP/IP) and with the DIS through the User Datagram Protocol/Internet Protocol (UDP/IP). The TCP/IP provides point to point communications between Janus and PASS and passes information that both Janus and PASS understand. The UDP/IP, on the other hand, broadcasts all PDUs of all the models involved in the exercise over the network.

During the PASS development, a research team determined that a minimum of three PDU types were necessary to visually model events that happen in the simulated world: the Entity State, Fire, and Detonation PDUs. [10] Appendix A shows the information included in each of these PDUs. Janus processes the information provided in the PDUs for its own internal reports and to physically represent the events with its own graphics. When Janus is operating in a distributed environment with either Janus or other

models, such as Modular Semi-automated Forces (ModSAF) entities, an equivalency file is required to translate an entity from DIS to one that Janus recognizes. The Equivalency Editor in PASS was built for this translation purpose. Any entity that cannot be found in the PASS Equivalency Editor will be ignored in Janus.

C. PROBLEM DESCRIPTION

1. Analysis Tools

Using models such as Janus in a distributed environment adds flexibility and enhances the analyst's ability to analyze various topics. Although, neither DIS nor HLA provide any analytical tools within their infrastructures, they do provide a framework to build tools and capture data to accomplish their purpose. In DIS analysts can formulate their requirements and questions before and after an exercise. They can then use post processor reports and data loggers. HLA, on the other hand, requires the analyst to determine the data requirements for his analysis prior to executing the simulation. Analysts can then perform their analysis by using post processor reports or a proposed Analysis Federate. The post processor reports are available after a simulation run is complete, whereas the Analysis Federate provides the additional capability of performing real-time analysis.

a. Post Processors

Post processors are commonly built into individual models. Therefore, if they are present in the model, their reports can be used in both DIS and HLA environments. Post processors provide various predefined report formats on specific events and measures of effectiveness (MOE). Although the specific reports vary from model to model, only the Janus post processor will be discussed here.

The Janus post processor prepares reports ranging from artillery impacts to ammunition expenditures to killer/victim scoreboards. Although most reports are strictly “bean counts,” a few provide calculated information providing measures of performance and effectiveness. A detailed summary of the available post processor reports is available in Appendix C.

To use the Janus post processor, the analyst selects the post processor option from the program execution menu for a specific scenario and run number. The user next specifies which reports he wants and the post processor creates a file containing them. The analyst then moves back to the program execution menu which allows him to either view the reports on the screen or print them to a printer. If the reports are viewed on the screen, the user must step through them line by line since Janus’ post processor reports still use relatively antiquated technology. Reports that are printed can be extremely long. For example, the detection reports can easily be over 100 pages long. In either case, these reports provide only limited information to the analyst.

b. Data Loggers (DIS)

Beside the post processor reports, an analyst in the DIS environment uses data loggers. Although DIS does not provide any analytical tools of its own, data loggers use the fact that PDUs are broadcast over the LAN or WAN to capture all of the PDUs and record all of the events and interactions that occur during the simulation. The data logger file becomes extremely large very quickly since much extraneous information is included. Since extracting the data manually could take hours or even days depending on the information needed, a program that can parse the file must be used.

The data logger file can be used by the analyst to gather data that are required to answer questions that may not be possible to answer by just using the post processors. Also, since models can interact from distributed locations, this file's collection technique allows an analyst that may not have a simulation model available to still collect information and analyze an exercise. For example, a commander with units split between various posts could set up an exercise over a WAN without requiring his unit leaders to gather at one test site. In this case, the command group could still gather the PDUs at their home station and conduct their analysis.

c. Analysis Federate (HLA)

As with DIS, HLA does not have any analytical tools available. One proposal to facilitate analysis in HLA is the Analysis Federate. [6] In the paper presenting the proposal, the Analysis Federate would create a SOM during its own development. However, in the actual implementation, the Analysis Federate parses the FOM and FED files and writes the Analysis Federate's SOM during the start up of the exercise. This implementation allows the Analysis Federate to interact in any HLA federation.

The Analysis Federate is a separate model that would interact with the federation in the same way a combat simulation model does. However, its purpose is strictly to gather and process information. The Analysis Federate subscribes to required data and does not publish unless it is specified in the FOM.

Since in the HLA environment a federate subscribes when it joins the federation, the analyst must predetermine his requirements. The Analysis Federate will only receive the objects, interactions, attributes, and parameters that it requests or

subscribes to. This means that only the information requested or subscribed to is available to the Analysis Federate for purposes of answering the study questions. Any additional data needed require the analyst to modify his subscription requirements.

With current technology, these data can be collected and processed simultaneously, allowing the analyst the option of conducting real-time analysis. The ease of conducting this analysis will be determined by the functionality of the graphical user interface (GUI) used.

2. Advantages and Limitations of the Analysis Tools Available

There are several advantages and limitations to three analysis tools listed above. First of all, the post processors are found in simulation models. Reports provided by the post processors have been developed over time by looking at past and potential future requirements. The information that is provided in these reports could be quite robust, as in the case of Janus. However, this list of reports is finite so, no matter how large the list is, someone is almost guaranteed to ask a question that is not covered in one of the provided reports. Additionally, although the post processor reports may contain information required to answer a specific question or MOE, it may not all be in one report. The analyst then would need to extract these data from the reports and manually process it. Furthermore, post processors only contain information that impact their specific simulation or can be converted to their model using equivalency editors. Other entities will not be included in the post processor reports. Therefore, although the post processor provides commonly used reports, it does not have the flexibility to alter the reports and may not contain all of the data needed.

Data loggers help bridge this gap in the DIS environment. The data loggers create files that contain all of the PDUs from an exercise. This gives the analyst the flexibility to answer questions that are not possible with just the post processor reports. With data loggers the analyst could develop, change, or modify his questions during the conduct of his analysis and still have the required data available as long as the data are included in the PDUs. However, the files storing the PDUs are extremely large and require long processing times to extract data. Additionally, bandwidth limitations can cause up to 15 or 20 percent of PDUs to be lost. This could impact greatly on the overall results.

The proposed Analysis Federate approach in HLA helps reduce this bandwidth limitation by sending only the information subscribed to by the federate. This subscription methodology could potentially greatly reduce the amount of extraneous information being sent over the network when compared to a DIS exercise. The Analysis Federate additionally adds the capability to conduct real-time analysis and provide immediate feedback to the analyst. The Analysis Federate, however, has limitations based on this reduced amount of data. Since data requirements must be determined before a federate can make its subscription requests in the federation, only questions pertaining to this information can be answered using the Analysis Federate. If the analyst decides that the data collection requirements must be changed during the execution of a simulation run, he will have to subscribe to the additional objects and interactions. After he makes these subscription changes, he will have no means of going back in time to collect the history of these new data elements. Instead, the answers to his analysis questions must be limited to the parts of the simulation that take after the new

subscription requirements have been processed. However, subscribing to all the data presents the same processing and bandwidth problems as with the data logger.

3. Analytical Approaches in DIS and HLA

Since the analytical tools are different in DIS and HLA, it seems logical that the approaches to answering questions would also be different. In DIS, analysts typically have a general idea of what they want to answer prior to conducting an exercise, and only when the exercise is over are they required to narrow the scope of their analysis questions to specific data requirements. The analyst can collect now and determine the requirements later. In HLA, however, the analyst must determine ahead of time exactly what is needed. A specific objective for each federation should be established prior to setting up the exercise. The analytical approach here, then, should take this objective and break it down into specific study questions, MOEs, and data collection requirements. In other words, HLA requires detailed planning prior to execution while DIS collects the data and sorts it out after the execution is completed.

D. THESIS STATEMENT AND OUTLINE

This thesis is a comparison between DIS and HLA's approaches to conducting analysis. The purpose is to determine functionality that is available in terms of gathering, processing and reporting on analytical questions in both environments. The emphasis is on how the questions are answered and not on the specific answers themselves.

The next chapter outlines the methodology used to compare the analytical approaches. To compare the different methods, three comparisons were conducted: Janus vs. Janus in DIS, Janus vs. Janus in HLA without an Analysis Federate, and Janus vs. Janus in HLA with an Analysis Federate. The next chapter also discusses the scenario

development, study question development, and the actual setup for the exercise. Chapter III presents the results of the different methods, giving the steps that were taken to collect, extract, and process the data to answer specific study questions for each method. Finally, Chapter IV provides recommendations and conclusions of the comparisons as well as potential for future work.

II. METHODOLOGY

DoD agencies and units regularly conduct exercises to maintain and increase the readiness of their forces, often combining live, constructive, and virtual simulations. This combination is termed Synthetic Theater of War (STOW). Exercises can range from platoon leaders through battalion commanders conducting an attack using Janus to a division level exercise incorporating forces distributed in tank simulators, computer simulated forces, and units in the field. Exercise development for each case is similar, the main differences involving the unit level of implementing the exercise. The development must consider several factors: available combat models, scenario, analytical questions and tools, software/hardware requirements, and the communications architecture.

This chapter outlines the methodology used in this thesis. The combat models, scenario, analytical questions and tools, and the software/hardware requirements are discussed in detail. By executing this exercise, analytical tools and approaches in both DIS and HLA can be scrutinized and an overall comparison of analysis techniques done.

A. COMPARISON OF ANALYTICAL APPROACHES

Three approaches will be compared: Janus vs. Janus in DIS, HLA without an Analysis Federate, and HLA with an Analysis Federate. Each approach looks at how analysts conduct their studies using the tools and techniques available or proposed for their respective environment. Both the approach methodology and analytical tools will be included in the comparison.

1. Janus vs. Janus in DIS

To conduct an analysis in DIS, two primary tools are available to the analyst: the individual simulation's post processor and the DIS data logger. The post processor will

include data from other non-Janus models as long as the entities are listed in the equivalency editor. Otherwise, the data will be lost and not processed when making the reports. As described in the previous chapter, both tools are used after the exercise is completed. Therefore, actual study questions do not need to be determined until the exercise is running or completed. Often the only requirement prior to the exercise execution is the purpose and overall objective of the study.

For this study, the post processor reports provided by Janus were used along with information extracted from the DIS data logger. The available post processor reports are listed in Appendix C. A software program developed for DOD that implements a data logger for DIS analysis is the Data Collection and Analysis (DCA) Tools. The latest version was released in February 1997 by its developers, Lockheed Martin. DCA Tools is essentially a general form of the individual model's post processors, providing a list of reports similar to those provided by Janus. However, due to the limitations of post processors as described in Chapter I, Section C. 2. and poor software documentation, DCA Tools was not used.

Instead, a PERL (Practical Extraction and Report Language) program was used to extract required information. PERL is a programming language that supports powerful text processing capabilities, including regular expressions. Since PDUs follow strict formatting rules, patterns are easily determined and can therefore be exploited to parse the data logger files. The resulting output can be put into a table format for easy reading or importing to another program, such as a spreadsheet, for further calculations and processing.

2. Janus vs. Janus in HLA without an Analysis Federate

Conducting analysis in HLA without an Analysis Federate demonstrated how an exercise can be analyzed using only the currently available tools in the HLA environment. Since HLA requires federates to subscribe when they register, current data loggers that passively listen to the network will not be of any use. To have a data logger that publishes and subscribes to all possible output would defeat the purpose of going to HLA in the first place. Additionally, the HLA infrastructure does not provide any analytical tools. This means the only tool available for this approach are the post processors that are already available in the individual models. Therefore this portion of the study will demonstrate how required data can be retrieved using just the post processor in Janus.

3. Janus vs. Janus in HLA with an Analysis Federate

The final comparison approach used HLA with the Analysis Federate. Again, due to the subscription requirements in HLA, an analyst must determine his requirements prior to executing the exercise. This means that more analysis is required up front to determine what the objective, study questions, MOEs, and objects and interactions are. This study question development methodology is discussed in section C of this chapter. Once all of this is determined, the analyst can subscribe to the appropriate data.

As discussed in the previous chapter, a proposed Analysis Federate has been developed. This Analysis Federate is a separate software package from the other federates that is used to subscribe to the required data in order to analyze the exercise. For this study, Vision XXI software from Tapestry Solutions was modified to become the GUI for the Analysis Federate. Additional capabilities were added in order to operate in

HLA and facilitate analysis. Also, the Analysis Federate is being developed for use in any federation. It will be discussed in more detail in subsection D. 1. in this chapter.

B. SCENARIO DEVELOPMENT

As with most combat simulations, a detailed scenario must be developed. A detailed scenario description should allow the players to understand factors the analyst or trainer are most interested in so they can understand how the underlying assumptions may affect the scope of their decision making. [3] The scenario description must consider several factors. Three of the most important factors are realism, flexibility, and tactical soundness. A scenario incorporating these factors will increase the training and analysis value of the exercise.

First, a realistic scenario motivates others to use the scenario and, once executing, to motivate them to feel as though they are participating in an actual battle. Realism can be achieved in various ways, such as using current events or hot spots, past battle scenarios, and future potential conflicts. An unbelievable and impossible scenario can negatively impact a trainee's motivation and, hence, the training value of the exercise. However, a motivated trainee will be more likely to act similarly to actually being in the situation for real and provide better data for analysis.

Next a scenario must be robust enough to allow the trainers and analysts some flexibility. This will give them leeway to modify or direct the exercise as necessary to meet specific objectives and goals. Many times the information required for a particular problem or exercise changes as the scenario continues. In order to handle these changes, a scenario must contain enough entities, events, and information output to allow the analyst the flexibility to answer or adjust his requirements. These events and entities

provide analysts a wide variety of information in order for him to make certain recommendations or conclusions.

Finally, the scenario must be tactically sound. Tactics encompass a wide variety of areas including fire and movement techniques and equipment characteristics. Doctrine described in field and other papers provides general guidelines and performance measures for conducting various tasks. It provides a base for setting up the tactics for the exercise. A tactically sound scenario, unless specified otherwise for analysis and training purposes, will provide the most accurate and useful results for the analyst.

C. ANALYSIS QUESTION DEVELOPMENT

The objective of the exercise provides the foundation in developing the scenario and analysis questions. The scenario and analysis questions must be developed closely together. The scenario provides the data for the analyst to answer his analysis questions. Therefore, the analyst must take into account what data are available from the scenario. In DIS, analysis questions are typically determined after a simulation run. On the other hand, HLA requires them to be determined prior to the exercise run due to the publishing and subscription requirements of registering with a federation.

Despite the difference in approaches, the study questions, and hence data requirements, must be the same in order to compare results and findings from this study. Since the study question development is stricter and requires more forethought in HLA, the HLA process was used to determine the study questions and data requirements. Once determined, these questions and requirements will be used for all of the comparison approaches.

As with DIS, a standardized analysis approach in HLA does not exist. However, a proposed HLA approach uses an “objective tree” to determine data and subscription requirements and answer study questions. [6] The objective tree starts with the exercise’s overall objective and works down through study questions and MOEs until the required data are determined. Then the subscription requirements can be determined from the required data.

First the objective of the exercise is determined. This could range from a training purpose to a weapon or tactical engineering purpose. Once the objective is determined, study questions are created that will allow the analyst to accomplish his objective. Each study question in turn may have other study questions based on them. This potentially could go on for a few or several levels.

Once the study questions are filtered down to the lowest required level for that objective, MOEs and data requirements are determined. If MOEs are used, they may involve equations and will be used to answer the study question under which they fall. Some questions may just require data. In either case, once the MOEs are determined, the final level of the objective tree is the required data. In HLA, this required data converts to the objects, interactions, parameters, and attributes that will be subscribed to when registering with the federation.

Figure 2 shows an example of how the study question model tree may look:

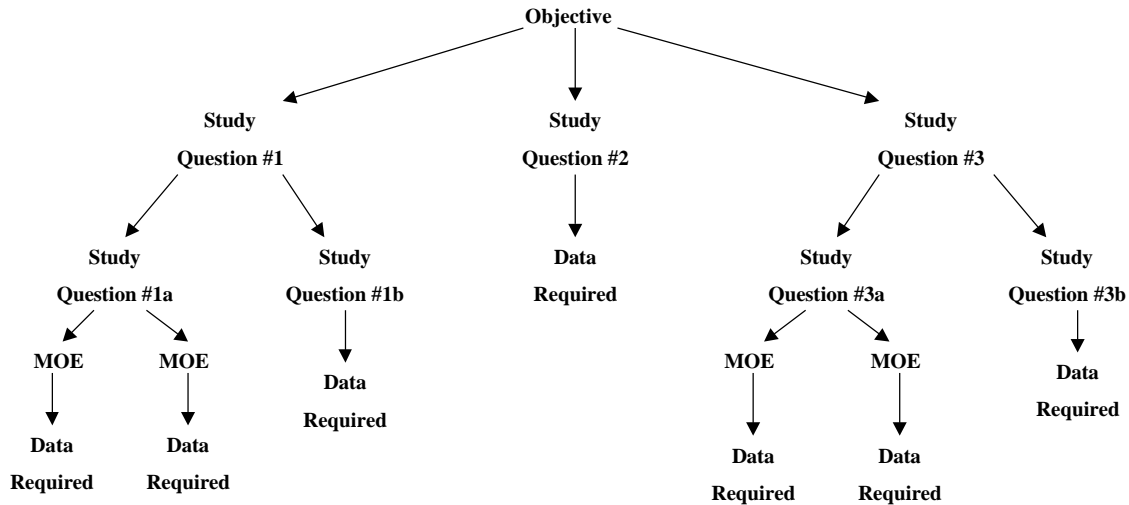


Figure 2: Study Question Model Tree.

The development of the tree goes from top to bottom. However, actually answering the questions involves going from bottom to top. Each level relates to the calculation or answer of the question above it. The outcome of moving back up the tree will be the objective.

D. SOFTWARE AND HARDWARE REQUIREMENTS

Software and hardware requirements can greatly impact an exercise. Despite cost, ease of use, and availability considerations, these requirements can severely limit or enhance the analysis to be done. For this particular exercise, the DIS requirements and environment are well established. However, HLA-compliant software, in particular Janus and PASS, is still in the early stages of development and full compliance will not happen for another year. This section will explain how the exercise analyzed in this thesis was setup in both environments.

1. Janus as the Base Model

Janus is going to be the base model for this study due to its proven and future potential and availability. By standardizing the model used, the entities, interactions, and

other model calculation algorithms will be the same for each simulation run. Although the results may differ due to random number seeds and time latency issues in DIS and HLA, the basic algorithms determining the outcomes will remain consistent.

2. DIS Setup

In order for Janus to operate in a DIS environment, Janus communication protocols are translated to DIS PDUs and vice versa using PASS. Janus operates on a Hewlett Packard 715/50 while PASS operates on a Silicon Graphics O2. Janus communicates with PASS using TCP/IP network protocols. This is a point to point transmission and is used to reduce transmission times and confusion if multiple Janus simulations are interacting.

DIS mandates that PASS communicate with other distributed models using the UDP/IP. The UDP/IP allows other models to retrieve the data they need as long as they are using the appropriate port number. This is where the data logger fits into the distributed environment. The data logger operates on a Silicon Graphics O2. It logs all of the PDUs being passed over the network. The log file produced can be used either to replay the battle or extract the data required to conduct analysis after the exercise is complete.

Figure 3 shows how Janus vs. Janus in DIS is set up for the exercise.

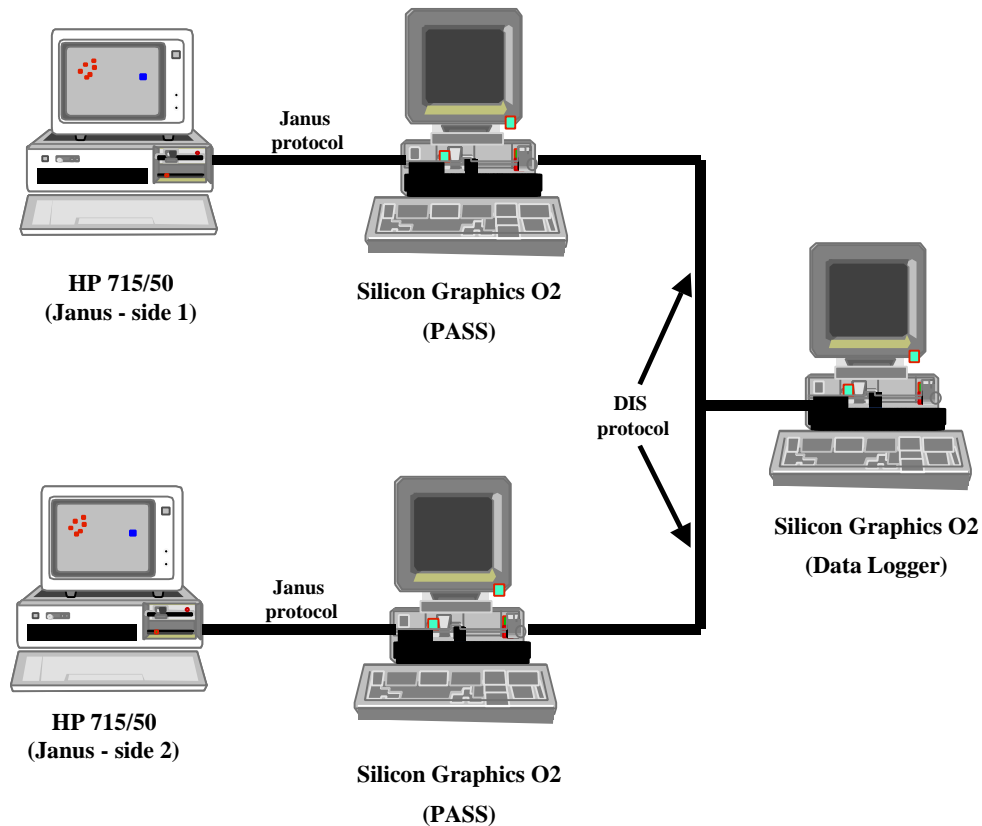


Figure 3: DIS Setup.

In total it takes two Hewlett-Packards and three Silicon Graphic computers to run this DIS exercise.

3. HLA Setup

In order to allow legacy models (developed prior to the acceptance of HLA as a mandated infrastructure) to operate in HLA, a short-term fix is the use of the Institute for Simulation and Training (IST) HLA Gateway software. This software converts the DIS PDUs into HLA compliant data formats. Also, the proposed Analysis Federate is being developed by TRAC-Monterey by adapting the Vision XXI software to work in an HLA environment and by extending the Vision XXI functionality. Both HLA Gateway and Analysis Federate software are discussed in the next subsections.

By using the short-term fix of the HLA Gateway software, a Janus exercise operating in HLA can be replicated on how a totally HLA compliant setup would look. Janus and PASS operate in the same manner as discussed above. However, now instead of the PASS models communicating directly, they send information through the HLA Gateway to the RTI back to the HLA Gateway to the other PASS model. The HLA Gateway software and the RTI operate on a Silicon Graphics O2. The Analysis Federate operates on a PC Solaris.

In this setup, Janus communicates with PASS using Janus protocol using TCP/IP. PASS converts the Janus protocol to DIS PDUs and communicates with HLA Gateway using UDP/IP. HLA Gateway then converts the PDUs to HLA objects, attributes, interactions, and parameters. The RTI, in turn, acts as a “traffic cop” by routing the requested information to the appropriate models. From this point the process just goes backwards from the manner just described: RTI to the HLA Gateway to PASS to Janus.

Figure 4 shows a diagram of the HLA setup.

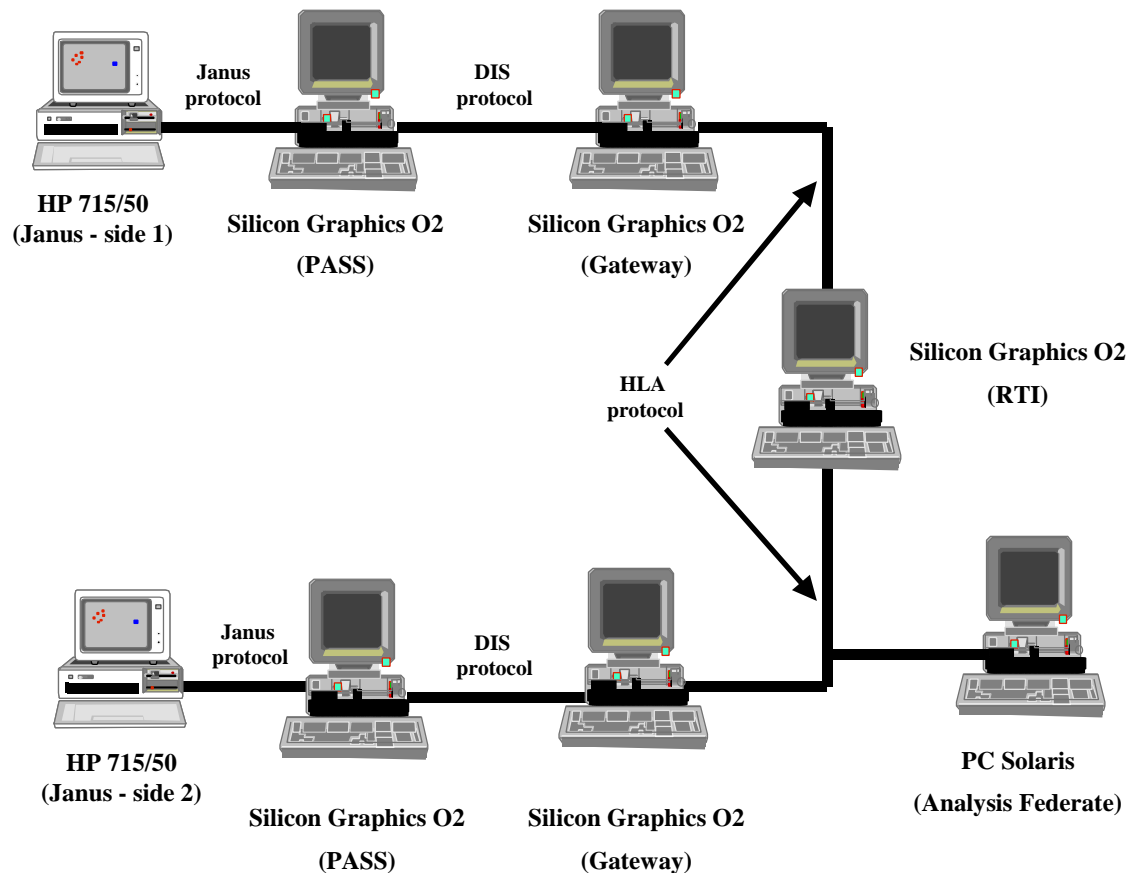


Figure 4: HLA Setup.

The PASS now broadcasts its PDUs to the HLA Gateway program. The HLA Gateway then converts the PDUs to HLA data formats. The information is then routed through the RTI and sent to the appropriate federates. The federation includes the Janus/Gateway federates, the Analysis Federate and the RTI. Together they create the federation.

All of these machines are on the same physical network. Logically there are five separate networks that force the data to be routed via the data flows depicted in Figure 4. These logical networks are implemented by using different port numbers between the individual machines. One reason these separate port numbers were required was to

prevent the two Janus simulations from communicating with each other via the two pass models using DIS protocols. Models have to be operating on the same port in order to transfer the simulation data. Information sent directly from PASS to PASS would skew the results in HLA.

a. HLA Gateway

The HLA Gateway program provides a means for legacy DIS models to interoperate in an HLA environment. It was written to work with the Real-Time Platform Level Reference (RPR) FOM. It is a stand-alone program that does not require any changes to existing models in order to operate in HLA. The RPR FOM is a Federation Object Model that was used to test HLA. It was written to handle the PDUs that are sent in DIS. The HLA Gateway receives the PDUs and performs two functions: it converts the data in the packets to the data formats indicated in the FOM and translates the sequence of the packets into the corresponding RTI service invocations. Additionally, the HLA Gateway's functions include creating, destroying, joining, and resigning federations and publishing and subscribing to RPR FOM classes. [12]

The Institute for Simulation and Training in Orlando, Florida developed HLA Gateway. The first versions of HLA Gateway were released in 1997 with the most recent version, 2.3, being released in March 1998. Appendix D details the modifications to the HLA Gateway software required to make it work for this exercise. Future versions are scheduled to be released later this year.

b. Analysis Federate

The Analysis Federate is a software package being developed by TRAC-Monterey that is designed to be used to perform analysis in HLA. [7] It uses the Vision

XXI software functionality developed by Tapestry Solutions in San Diego, CA. Vision XXI was originally developed to be used as a command and control tool for commanders and staffs to analyze a battle to facilitate decision making. TRAC-Monterey adopted the software package to work as an Analysis Federate graphical user interface. The Analysis Federate is designed to work with any HLA federation. This provides a flexible and reusable tool for analysts to use in the HLA environment.

The Analysis Federate first parses the FOM and FED file from the RTI to determine the data formats, objects, interactions, attributes and parameters that are available. These are then translated to a graphical user interface (GUI) that allows the analyst to publish and subscribe to the required data. Once the analyst has selected his data requirements, the Analysis Federate joins the federation and is ready to begin collecting data.

In addition to the registration GUI, the Analysis Federate provides a GUI for analyzing the battle that can be operated during runtime or after the battle is complete. It provides a map sheet that shows characteristics about entities such as their movements, strength, if they are in combat, and numerous other information. Various reports are available ranging from breaching operations to unit strength to indirect fire missions. Additionally, the analyst can select the particular time frame to observe and the type of unit to analyze. Reports from Vision XXI can be saved as a file to be read later or imported into another program, such as a spreadsheet, for further calculations and processing.

In summary, DIS and HLA are two different approaches towards facilitating interoperability between various models. As such, both provide a different

environment to conduct analysis. Although post processor reports are common to both in the individual models or federates, analysts must rely on the environment specific informal HLA and DIS analysis tools to accomplish their objective in distributed environments. The theoretical differences of these analysis tools were discussed in this chapter. The exercise results are discussed in the next chapter.

III. RESULTS

The methodology outlined in Chapter II provides the theoretical approach to a comparison of analysis conducted in DIS vs. HLA. To more fully understand analysis in both environments, an exercise was executed which included all the phases of development from the scenario development to answering specific analysis questions. The results are discussed in this chapter.

First, the scenario used for this study is discussed. It incorporates the important factors of scenario development with the strengths and limitations of the Janus combat model. The scenario is based on a hypothetical future conflict set in Bosnia. Next the analysis questions for this study are developed and refined into MOEs and data collection requirements using the methodology in Chapter II. Finally the results from each type of analytical approach are given to each of the analysis questions and an overall discussion of the results is presented.

A. SCENARIO DEVELOPMENT

1. Exercise Scenario

The following scenario provides the general background, current situation, and most recent events leading up the start of the simulation run. It is fictitious but incorporates the ideas described Chapter II, section B. In practice this scenario description could be read to those involved in the exercise to focus them and set the stage for what was about to happen. Additionally, it could be used to establish a guideline for the analysts and simulation controllers to use in representing the scenario in the combat models.

a. General Situation

After four and a half years of conducting convoy security, civil affairs, and nation building missions in Bosnia, the United States finally declared success. The U.S. claimed the Bosnian government was stable enough to begin a U.S. withdrawal. The plan was to withdraw task force size units out of the port of Dubrovnik on the Adriatic Sea. In order to maintain political, economical, and military stability, these redeployments would occur staggered over an eight-month period.

Dubrovnik was chosen as the debarkation port because of its location and port facilities. Although the mountainous terrain restricts most off road movement along the way, the U.S. forces could move from Tuzla along the southern border of Bosnia to the port while staying out of site and mind of the military factions. Most road networks through Bosnia are limited to north-south with only a few going east-west due to the mountains. The few existing east-west roads are instrumental to maintaining the cross-country movement of humanitarian aid and other supplies to Sarajevo and other eastern towns in southern Bosnia.

b. Specific Situation

Up through the fifth month of withdrawals, the U.S. had redeployed about 70 % of their forces. Everything had gone well so far with only a couple of minor incidents during the movements. However, unknown to the U.S., Serbians were still upset over the land distribution from the Dayton Peace Accords and further infuriated by the U.S.'s arrogance during their missions in Bosnia. In an attempt to "teach" the U.S. a lesson, Serbian forces were massing units along the border approximately 60 km east of Gorazde. Their plan was to attack and secure the southern portion of Bosnia. In their

view, this not only would embarrass the U.S. but also would give Serbia a route to and control of a warm water port (Dubrovnik).

The Serbians initiated their attack at the beginning of the sixth month of U.S. redeployment. The U.S. still had a brigade in Tuzla and a battalion task force in Dubrovnik. The Serbians planned on rapidly moving to Dubrovnik and destroying the understrengthed and unsupported U.S. forces there while setting up defensive positions at decisive locations along the southern route. However, a quick counter attack by U.S. and United Nations forces from the north caused the Serbians to change their plan. Serbian forces halted their movement south and west and set up defensive positions at key road intersections along the southern route. A Serbian company team was able to reach the vicinity of Gacko before being forced to halt and defend the captured ground. Their mission was to block any forces moving north in an attempt to strike the Serbians.

The U.S. battalion task force located at Dubrovnik was in the final stages of preparing their vehicles for redeployment when the Serbian attack occurred. Therefore they were unable to move north for about a day and a half while reconfiguring their vehicles. This gave the Serbian force at Gacko approximately 30 hours to prepare their defenses. The U.S. force consisted of a balanced task force of 2 tank companies, 2 Bradley Fighting Vehicle (BFV) companies, an artillery battalion, and an engineer platoon. They still had their basic load of ammunition and communications with the commander of U.S. forces, Bosnia, located in Tuzla.

c. Current Situation

U.S. and U.N. forces in the north successfully halted the Serbian attack and pushed them back to the vicinity of Visegrad. The Serbians were starting to prepare

defensive positions along the southern route in order to hold the ground they had captured. The U.S. and U.N. forces were beginning to move additional forces to Bosnia to reinforce those already there. Additionally, an aircraft carrier task force already in the Mediterranean Sea was moving towards the Adriatic Sea to support the U.S. attacks. Additional aircraft were being deployed from U.S. bases in Europe and the United States. Also, a Marine Expeditionary Unit (Special Operations Capability) [MEU(SOC)], recently training in Spain, had landed in Dubrovnik and was preparing to attack north along the southern route.

The U.S. task force in Dubrovnik had moved north and was in position to attack the Serbian company team in the vicinity of Gacko. Their mission was to attack no later than 0600 hours to destroy Serbian forces in the vicinity of Gacko in order to gain control of the road network and pass the MUE(SOC) north.

2. General

The scenario developed for this exercise incorporated the factors discussed in Chapter II: realism, flexibility, and tactical soundness. Although these factors do not make up an exhaustive list, they are three of the most important ones. The goal is to build a viable scenario that will produce realistic and workable results.

As the scenario location Bosnia was chosen because of the recent conflict and current U.S. involvement. Ever since U.S. forces deployed to Bosnia in 1995, many U.S. politicians and citizens have been lobbying for the return of our forces. Finally in March 1998, the Army Times reported that “amid increasing optimism over peace prospects for Bosnia, the Clinton administration is planning to reduce the number of American troops

attached to Bosnia's NATO mission from 8,500 to 7000 this year." [13] This reduction is projected to begin later this year.

By using a scenario that has been in the news and on one that has been extensively reported and speculated about recently, combat leaders and analysts alike will be motivated to work towards the optimal solution for solving the conflict. Once the simulation has begun, the scenario will motivate the "players" by making them feel as though they are actually participating in a real battle.

The terrain database for Bosnia for this scenario was not available, so the Bosnian terrain was replicated by using a terrain database for Fort Hunter-Liggett, CA. The database represented the mountainous terrain of Bosnia with terrain that restricted movement mainly to north-south with only few east-west routes. Cross county movement is also restricted due to the mountainous terrain forcing most travel to the roads.

Realism can also be enhanced by using entities that are currently in the inventory of both sides as well as the standard number of vehicles for both sides. For this scenario, the following entities were used: tanks, infantry-fighting vehicles, engineer vehicles, artillery pieces, and obstacles. The actual breakdown according to each unit is given in Appendix D.

Entity type, number, and employment provide flexibility for the analyst to design a scenario to meet his specific objectives. For this scenario, U.S. forces attacked a hastily defended opposing force. Battle events included artillery missions, obstacle encounters, and tactical movement formations and engagements. These events and entities provide a variety of potential output and analysis opportunities for the analyst.

Figures 5 and 6 are snapshot pictures of sides one and two at the beginning of the battle. Side one represents U.S. forces while side two represents Serbian forces.

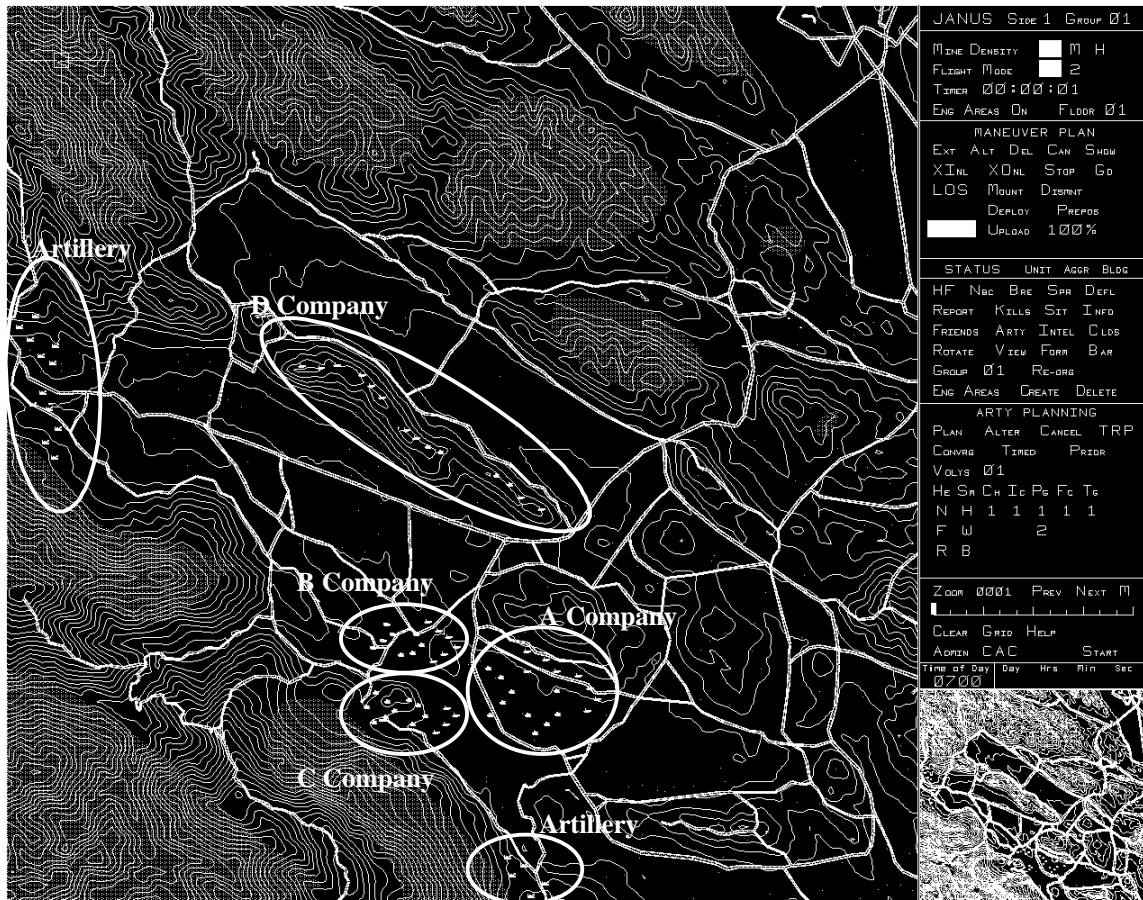


Figure 5: Initial U.S. Force Disposition.

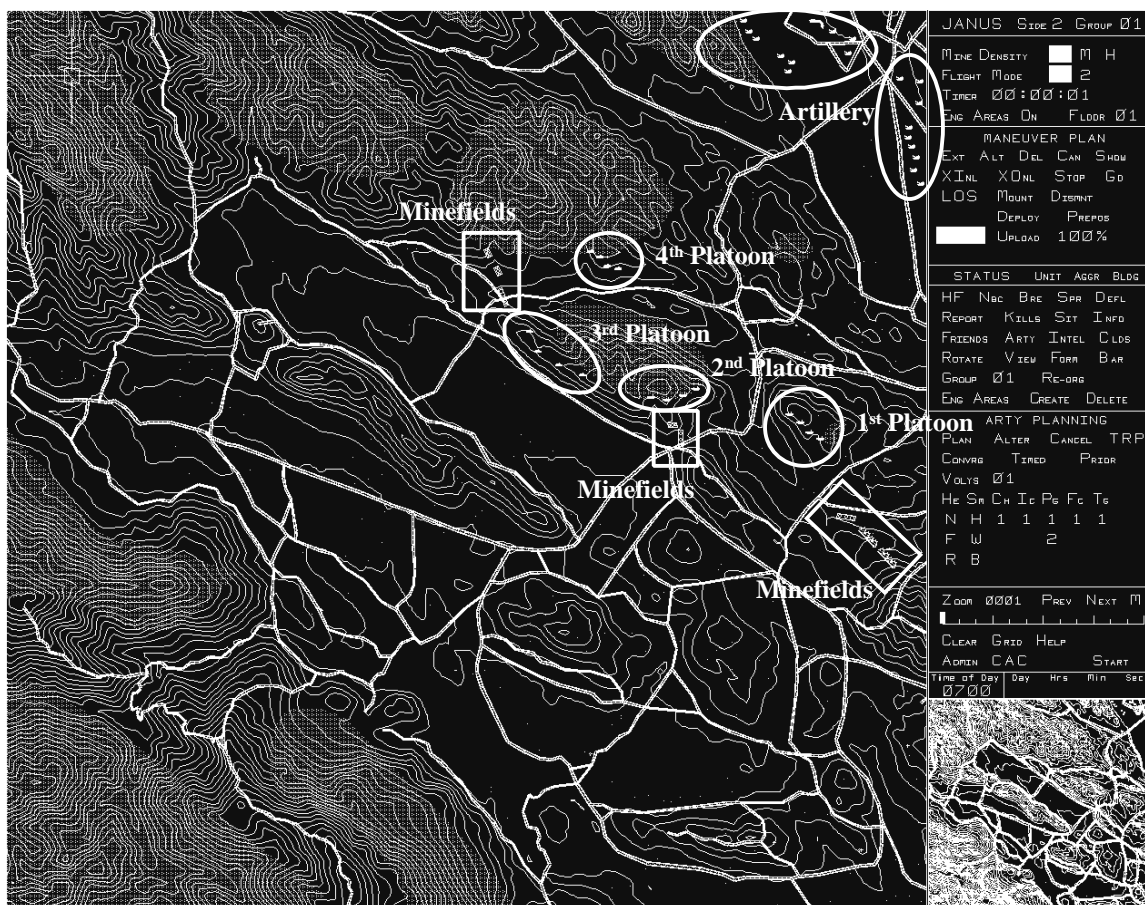


Figure 6: Initial Serbian Force Disposition.

Side one forces are in their assault and support by fire positions ready to commence their attack. Side two forces are in the defense preparing their fighting positions and obstacle plan. Since for this scenario side two has only had about 24 – 30 hours to prepare their defenses, the scenario was set up to replicate their preparedness level. In other words, their vehicles were not dug in yet and a limited number of obstacles were in place.

The final factor that was considered for this scenario was tactical soundness. A tactically sound scenario adds realism and increases the value of the results. U.S. doctrine calls for a preferred ratio of three to one when conducting an attack. This means three attacking entities to one defending entity. In this scenario, there are 56 tanks and

infantry fighting vehicles attacking against 16 equivalent type vehicles. This is a 3.5 to 1 ratio. The two sides are equal in the number of artillery pieces, a battalion of 18 howitzers. Additionally, the U.S. forces have 4 engineer vehicles. Overall, the U.S. forces have a total of 78 vehicles while the opposing forces have 34.

The next part of doctrine to consider is the tactical movement and formations employed during the attack. Normally a support by fire position is established in order to fix or suppress the enemy while the maneuver forces conduct their attack to destroy the enemy. In this scenario, two tank and one infantry fighting vehicle (IFV) platoons from D Company suppress the enemy while three company teams, a mix of tank and IFV platoons, move in platoon and company wedges to conduct the attack. The Serbian forces, on the other hand, deployed their forces in an attempt to maximize their effectiveness in controlling east-west road movement. Overall, the scenario gives a starting point for the analyst to begin his study.

B. ANALYSIS QUESTIONS

With a detailed scenario now established, the analysis questions can be developed. The proposed approach (HLA with the Analysis Federate) requires these questions, MOEs, and data collection requirements to be determined prior to the beginning of the exercise. The objective is used as the starting point and the data requirements as the finishing point. The rest of this section shows the analysis questions developed for this study.

First, the overall objective for the exercise must be determined. This should actually be determined before the scenario is developed, since it provides the purpose and framework required to set up and analyze the exercise. For this exercise, the objective is

to increase the tactical decision-making ability of the leaders from the platoon to battalion task force level. All questions and actions taken in this exercise reflect this objective.

Given the objective, the following study question tree was developed.

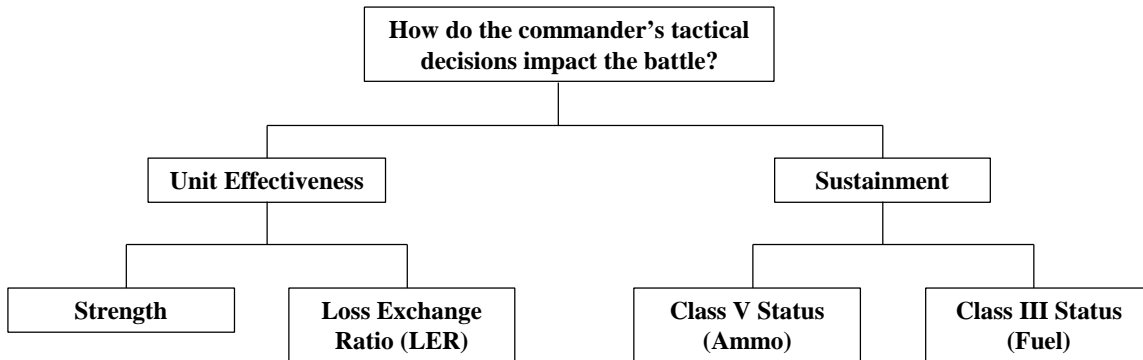


Figure 7: Study Questions.

One question that would help in achieving the overall objective for this exercise is to look at how the commander's decisions impact on the battle. Two areas to help answer this are the unit's effectiveness and their sustainment during and after the battle. In turn, unit effectiveness can be broken down into percentage strength remaining and the loss exchange ratio (LER). Similarly, sustainment can be further divided into class III (fuel) and V (ammo) status. By analyzing each of these areas and working back up the "tree," the impact of the commander's decisions can be analyzed and determined.

To determine the unit's strength and LER and the fuel and ammo status, additional MOEs and data collection requirements were developed (see Figure 8). The unit's strength takes into account both the starting number of units or entities in each and the remaining numbers at specific times throughout the battle.

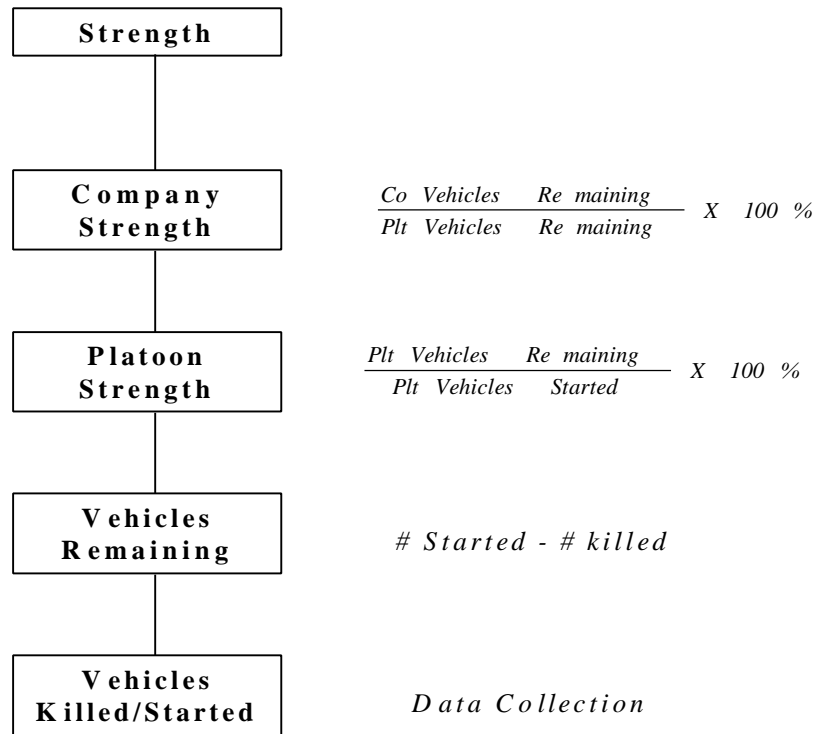


Figure 8: Unit Strength MOEs and Data Collection Requirements.

By collecting the number of vehicles each unit started and ended with, the vehicles remaining can be determined. The platoon and company strength MOEs use these numbers to determine their overall percent remaining.

Next, the unit's LER requires a killer-victim scoreboard, that is, which vehicle killed which vehicle. From this, the total vehicles killed and those killing them can be aggregated into their appropriate higher unit, as shown in Figure 9.

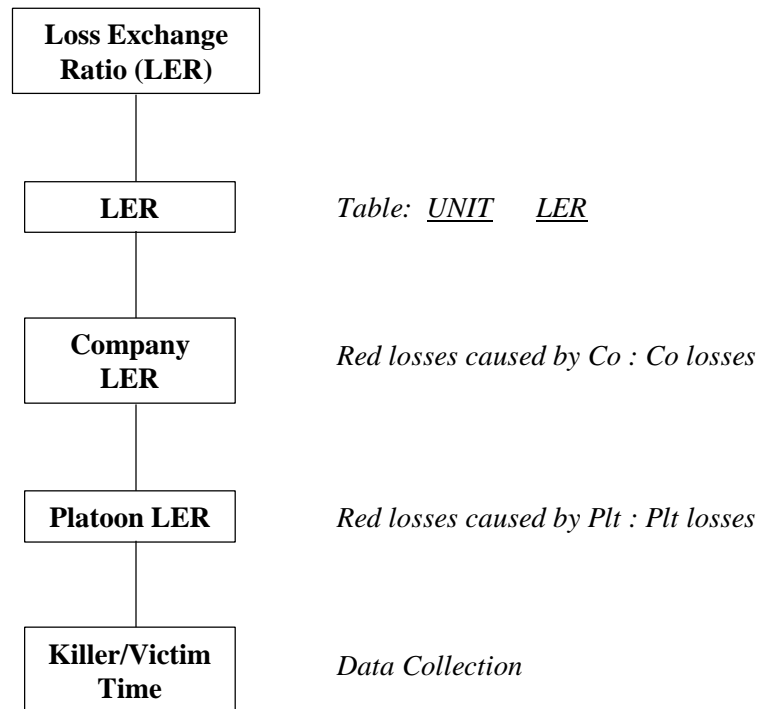


Figure 9: Unit's LER.

Once the unit's LER and strength are determined, the overall status of unit effectiveness can be determined.

Both fuel and ammo statuses are determined in similar manners. Each status requires the collection of the basic load, or starting fuel level, and the amount expended at specific times throughout the battle. Figures 10 and 11 show the breakdowns for both.

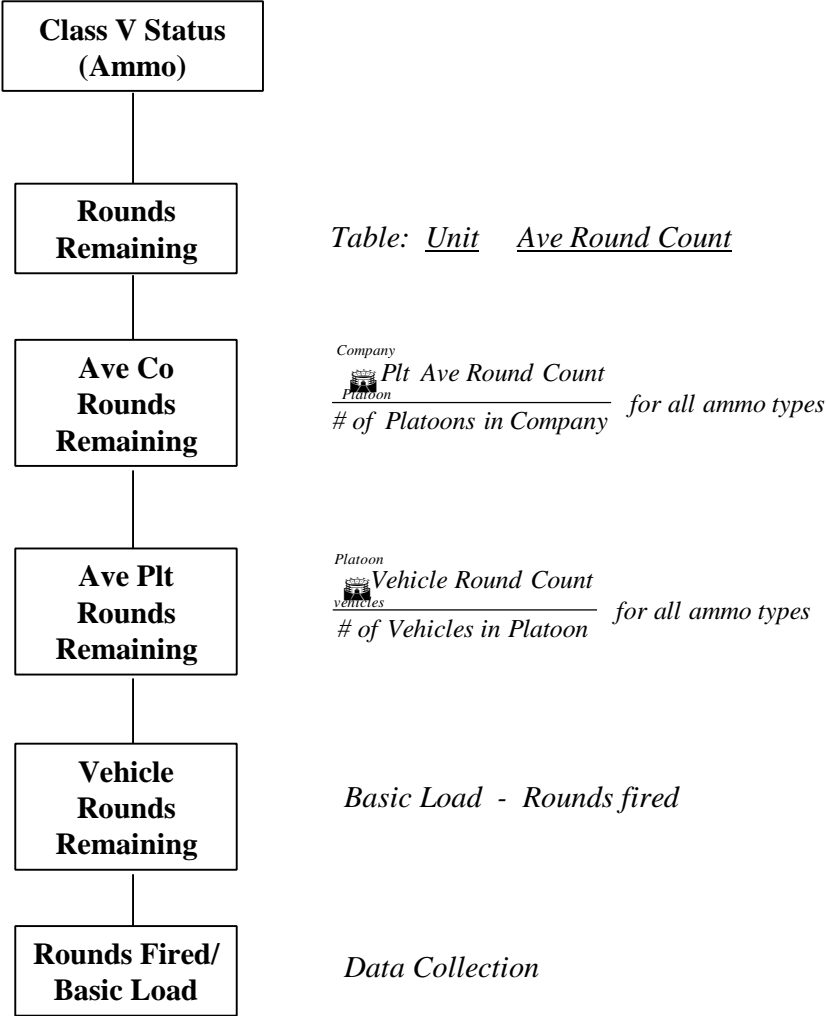


Figure 10: Class V (Ammunition) Status.

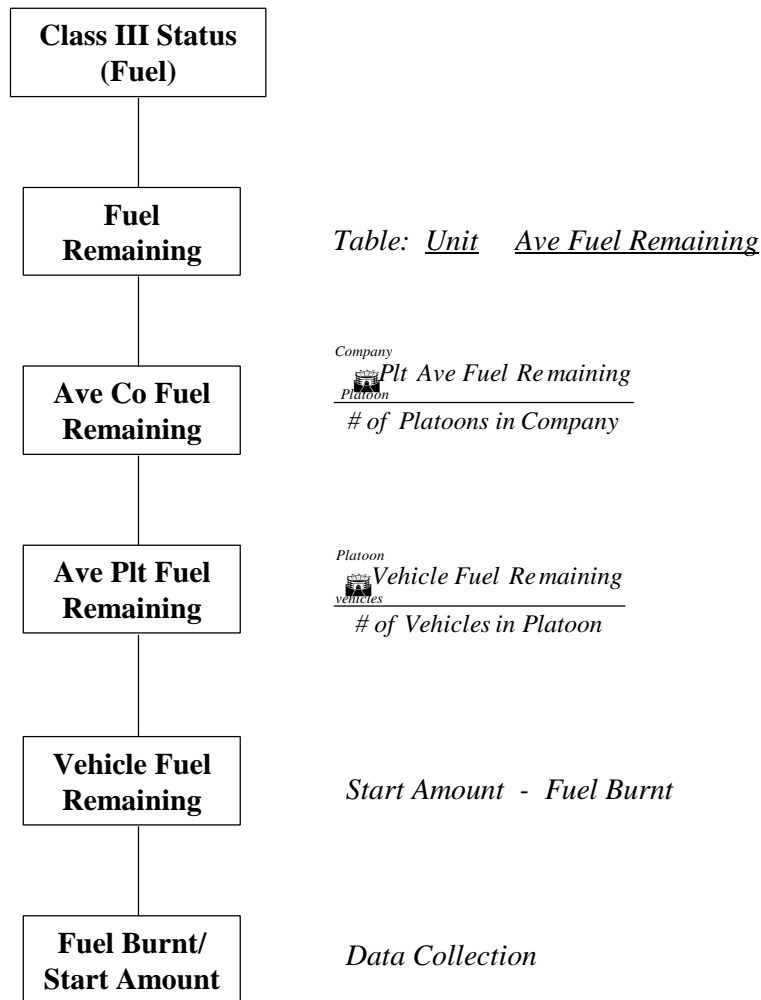


Figure 11: Class III (Fuel) Status.

Once the fuel and ammo statuses are determined, an overall assessment can be made on the unit's sustainment level.

Now that the unit's effectiveness and sustainment level have been determined, the analyst is ready to answer the first study question on how the commander's decisions impact the battle.

C. COMPARATIVE APPROACHES

Now with the scenario built and the objective and analysis questions developed, the exercise was run using the three different approaches discussed in Chapter II: Janus vs. Janus in DIS, HLA without an Analysis Federate, and HLA with an Analysis Federate. All three of the approaches used their respective infrastructure and analytical tools to execute the exercise and answer the questions.

1. Janus vs. Janus in DIS

a. Data Collection

During the DIS run, PDUs were captured using a data logger that allowed the UDP/IP port and exercise number to be specified. Any PDUs sent over this port were captured by the data logger and stored in a binary file. At the end of the exercise run, the data logger was stopped and the file saved. The data logger file for this exercise was over 20 MB in binary form.

Next, the data file was transformed into ASCII in order to run the PERL programs discussed in Chapter II, section A. 1. In order to convert the file from binary to ASCII, the data logger file was replayed over the UDP/IP. On a separate machine, a script file was started to record anything written to the screen and a PDU Dump program was executed that grabbed the PDUs sent over the specified port and printed them to the screen. Once the data logger had sent all of the PDUs, the PDU Dump program and script file were stopped. The script file now contained all of the PDUs for the exercise in ASCII format. This conversion caused the data logger file to go from over 20 MB in binary format to 93 MB in ASCII format.

With the data now collected and stored, the required data were extracted from the data logger file using two PERL programs. The first program parsed the data logger file to find out what vehicles were killed by other vehicles and when. This is essentially the same as the killer/victim scoreboard common in post processor reports and analysis. This program provided the required data for the strength and LER analysis questions. The other program parsed the data logger file to find out what rounds were fired and by whom. This program provided the required data to answer the ammunition sustainment analysis question. The fuel sustainment data requirements were unattainable in DIS and will be discussed in more detail later in this section.

For the first program, two PDUs are of interest: Detonation and Entity State PDUs. After an entity fires a round, the model controlling that entity determines when and where the round will impact. At the appropriate time, then, the model sends out a Detonation PDU. The Detonation PDU contains information on the firing entity, target entity, game time of detonation and the detonation result. The detonation result specifies if the round impacted the ground or target or exploded in the air. Rounds that impact the ground signify either misses or artillery rounds. Target hits signify direct fire hits. Rounds that explode in the air are typically from surface-to-air missiles and air-to-air missiles and were not used in this scenario.

If a round impacts a target, the model controlling the target entity determines what damage occurred. This information is then passed to the other models in an Entity State PDU. The Entity State PDU contains the entity's identification, game time, and damage assessment. If the target was destroyed, the entity's Entity State PDU will show *Appearance_DamageDestroyed*. The Entity State PDU will continue to show

that the entity was destroyed until the end of the simulation. If an artillery round destroyed a target, there is no way to tell in DIS.

From the time a Detonation PDU is sent until the time the damage assessment is sent in the Entity State PDU, anywhere from several up to hundreds of other PDUs may have also been sent. Also, it may take the model controlling the target entity a while to determine and send the damage assessment in an Entity State PDU. Therefore, just because a ground-impact Detonation PDU is followed by an Entity State PDU for the target entity, the actual damage may not be assessed until later.

The first PERL program creates a killer/victim scoreboard by reading ASCII files line by line and making the appropriate checks. First the code looks for a Detonation PDU. Once it finds one, it temporarily stores the game time, firing entity, and target entity until the detonation result can be checked. The result is listed at the end of the PDU. If the result is *DetResult_EntityImpact*, the temporarily stored data is stored permanently in a hash table. Otherwise, the temporarily stored information is erased. The actual code is in Appendix E.

Next, the program searches for an Entity State PDU. Once found, the entity is checked with all of the target entities listed in the hash table to see if it was hit by a round. If the target entity matches one of the listings in the hash table, the program checks to see if it is destroyed by looking for *Appearance_DamageDestroyed*. If found, the target's status in the hash table is updated to reflect that the target is destroyed. Otherwise, the program continues without updating the target's status.

After all the lines of the ASCII file have been read and processed by the program, it outputs a table with the target entity, firing entity, game time, and status for

all the destroyed entities for both sides. The information is stored in a text file for use later.

The second PERL program extracts which entity fired what round. This program is only interested in the Fire PDU. The Fire PDU contains the firing entity and the munition entity type. In DIS, any time an entity fires, a Fire PDU is sent by the model that controls that entity. One Fire PDU is sent for each round fired.

Once a Fire PDU is found, the firing entity identification and munition entity type are temporarily stored and a check is done to see if that entity has fired that type of round previously. If it had, the number of rounds fired of that type are incremented by one. Otherwise, the entity and munition types are added to the rounds fired hash table. Once all the lines of the ASCII file have been read and processed by the program, the program outputs a table with the firing entity identification, round type, and the number of rounds fired. The actual code is in Appendix F.

b. Results

Once the programs described in the previous section have parsed the data logger file, calculations and processing can be performed to determine answers to the MOEs and analysis questions. Table 1 shows a sample of the output from the killer/victim scoreboard program.

<u>Target Entity</u>	<u>Firing Entity</u>	<u>Game Time</u>
Site 57, Host 51, Entity 51	Site 58, Host 38, Entity 8	1118.2400 sec
Site 57, Host 51, Entity 15	Site 58, Host 38, Entity 11	964.2400 sec
Site 57, Host 51, Entity 34	Site 58, Host 38, Entity 19	1513.9600 sec
Site 57, Host 51, Entity 16	Site 58, Host 38, Entity 11	862.3000 sec

Table 1: Killer/Victim Scoreboard.

The entities are identified by the site, host, and entity number while the game time is given in total seconds from the start of the simulation run. The site and host identify which machine controls the entity. The entity number identifies the exact entity within the model.

The PERL program that determines the ammunition expenditure outputs a similar table. Table 2 shows a portion of this output.

<u>Entity</u>	<u>Round Type</u>	<u>Count</u>
Site57,Host51,Entity51	Kind2 Dom2 Cntry225 Cat2 Scat6 Spec5 extra0	2
Site57,Host51,Entity70	Kind2 Dom9 Cntry225 Cat2 Scat14 Spec2 extra0	22
Site57,Host51,Entity35	Kind2 Dom2 Cntry225 Cat1 Scat1 Spec2 extra0	7
Site57,Host51,Entity71	Kind2 Dom9 Cntry225 Cat2 Scat14 Spec2 extra0	3

Table 2: Round Count by Entity.

Entities are identified as described above. Round types are identified by the DIS enumeration for each round.

Although these tables contain the information needed, the entity numbers in Janus do not match the numbers in the data logger file. The entity and round types must be translated to specific vehicles within a platoon and company. To convert the entity types, first each entity in Janus should be assigned a platoon and company. Next, these entities need to be mapped to the site, host, and entity numbers used in DIS and the data logger file. To do this, a script file must be used when starting each PASS. This script file will capture the site, host, and entity numbers PASS assigns for the new entities coming from other models. PASS assigns these numbers in the same order as the entities are sent. Janus also assigns entity numbers in the same order they are received. By opening the unit data file in Janus' post processor directory for the specific scenario and run number, the order of entity assignment can be determined. Now the site, host, and

The round types can be determined more easily. PASS maintains an equivalency editor that matches DIS enumerations to specific entity types. This file contains a subsection for munition types with their enumerations. By simply comparing the DIS enumerations given in the PDU with those given in the equivalency editor, each enumeration can be translated into a specific round type.

By importing the output files from the PERL programs into Excel and translating the entity types and enumerations to specific vehicles and rounds, MOE calculations and answers to the analysis questions can be determined. Each of the analysis questions listed in section B were answered using the PERL script output tables imported into Excel. The A Company results will be shown in this section.

The first question dealt with the unit strength remaining. Table 3 shows the resulting output from calculations and processing done using the PERL output from the killer/victim scoreboard.

Company	Platoon	Start #	Killed #	End #	% Remaining
A	1	4	2	2	50.00
	2	4	4	0	0.00
	3	4	2	2	50.00
	4	6	5	1	16.67
	total	18	13	5	27.78

Table 3: Percent Remaining for each Platoon in A Company.

The low percent remaining is reflected in the fact that A Company was the lead company in the attack against the Serbian forces.

The next question dealt with the Loss Exchange Ratio (LER). The LER is the ratio of those entities a unit killed to the number of losses the unit suffered. Table 4 shows the LER for A Company.

Company	Platoon	Losses	Kills	LER (kills:losses)
A	1	2	1	1:2
	2	4	1	1:4
	3	2	0	0:2
	4	5	0	0:5
	total	13	2	2:13

Table 4: A Company's LER.

For A Company, the ratio is better if the number of kills is larger than the number of losses. In this case, A Company's LER is very poor. The poor showing is once again reflected in the fact that A Company was the lead element in the attack.

The ammunition expenditure MOEs and analysis questions use the basic load and the rounds expended information. The ammunition expended is found in the output from the PERL code. However, the basic load information for each vehicle was found in Janus' database. Table 5 lists the ammunition expended by A Company by round type.

Company	Platoon	Round Type	Start #	# Fired	End #	% Remaining
A	1 (M1A1)	tank rnd 1	112	1	111	99.11
		tank rnd 2	48	0	48	100.00
		AP round 2	4000	0	4000	100.00
		Lt armor 1	6000	0	6000	100.00
	2 (M1A1)	tank rnd 1	112	2	110	98.21
		tank rnd 2	48	0	48	100.00
		AP round 2	4000	0	4000	100.00
		Lt armor 1	6000	0	6000	100.00
	3 (M2 IFV)	missile 19	48	0	48	100.00

4 (M2 IFV) Co	AP round 2	2000	0	2000	100.00
	Lt armor 1	2000	0	2000	100.00
	missile 19	48	0	48	100.00
	AP round 2	2000	0	2000	100.00
	Lt armor 1	2000	0	2000	100.00
	tank rnd 1	224	3	221	98.66
	tank rnd 2	96	0	96	100.00
	missile 19	108	0	108	100.00
	AP round 2	13000	0	13000	100.00
	Lt armor 1	17000	0	17000	100.00

Table 5: Ammunition Expenditure.

From this table, A Company and its platoons did not expend many rounds. Since only 27% of its vehicles remained at the end of the battle, many of the vehicles most likely were killed before they got a chance to fire.

The final analysis question is similar to the previous one. Instead of ammunition expenditure however it deals with fuel consumption. The DIS PDUs did not include fuel consumption information. Therefore the MOEs and analysis question answers could not be determined.

Summarizing the information attained for each of the analysis questions, an overall analysis of the exercise's objective can be made. For this case, A Company did not do well. They lost 73% of their forces (13 vehicles) while only managing to kill 2 of the enemy's vehicles. For sustainment, although having most of their ammunition remaining, only 5 of their vehicles remained, so remaining ammunition is seen as a misleading statistic. Evaluating all of this information, A Company needs to reassess their tactics and determine if better decisions could be made in conducting this attack.

2. Janus vs. Janus in HLA without an Analysis Federate

The exercise for Janus vs. Janus in HLA without an Analysis Federate was run similar to the way it was described in chapter 2, section D. 3. However, the PASS and Gateway programs were run on the same machine reducing the number of computers required to run this scenario in HLA. Figure 12 shows how the exercise was set up.

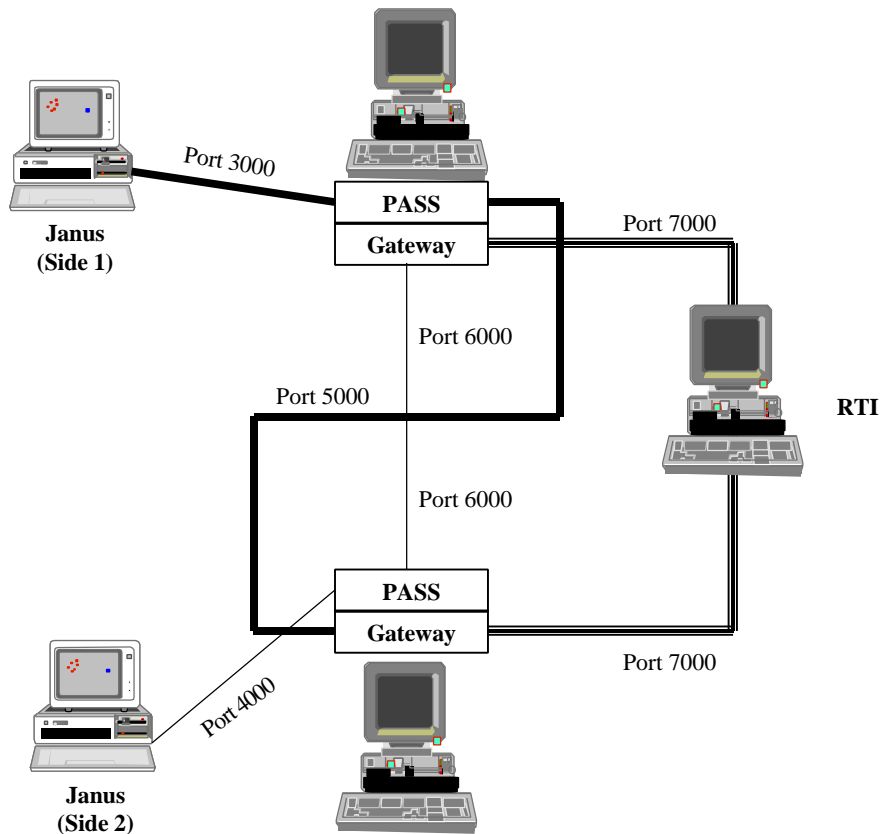


Figure 12: Janus vs. Janus in HLA without an Analysis Federate.

Since Gateway takes DIS protocol and converts them to HLA attributes, objects, interactions, and parameters, different port numbers were used to ensure that information was not passed between the models with the DIS protocol. However, due to the PASS system design, PASS and the Gateway could not communicate. Instead, the PASS on

one SGI would communicate with the Gateway on the other computer. As a result, the number of computers required was reduced from the original requirement of eight down to six.

a. Data Collection

Using this setup, the scenario developed earlier was run. The only tool available was Janus's post processor reports. Since the types of reports are limited, so are the available data. The Artillery Impacts, Direct Fire, and Coroner's Reports are required to answer the analytical questions for this study. The Killer/Victim Scoreboard, Artillery Summary, Direct Fire Ammunition Expenditure, and Force Loss Analysis Reports only contain aggregated information for all the entities for each side. Therefore, they are not of any use to gather data on sub-units.

As with DIS, only three of the four analysis questions could be answered in HLA, since fuel consumption could not be estimated for this exercise.

To gather the data required for the first two analysis questions, the only report that is necessary is the Coroner's Report. This report provides the game time of kill, the killer and victim's unit identification, and the location of the kill. These reports, as with all of the post processor reports, can either be viewed on the screen or sent to a printer for a hard copy. When viewing the reports on the screen, the return key is hit in order to scroll through the report. Once a line of the report is off the screen, the only way to go back is to restart the report. For either way of viewing the data, the killer and victim data has to be extracted manually and entered into the spreadsheet.

For the next analysis question, ammunition expenditure, two reports are needed: the Artillery Impacts and Direct Fire Reports. The Artillery Impacts Report lists

each indirect fire round fired for both sides and the Direct Fire Report lists each direct fire round fired. Again the data was extracted manually and entered into the spreadsheet.

b. Results

Once the data was entered in Excel, the remaining calculations and processing was the same as in DIS. The following results are shown for A Company. The other unit's results were determined in the same manner.

Table 6 shows the strength percentage remaining for A Company.

Company	Platoon	Start #	Killed #	End #	% Remaining
A	1	4	4	0	0.00
	2	4	2	2	50.00
	3	4	4	0	0.00
	4	4	3	1	25.00
	HQ	2	1	1	50.00
	total	18	14	4	22.22

Table 6: A Company's Strength.

As with the DIS simulation run, A Company was the lead element and received the most casualties in the battle.

Table 7 is the LER for A Company.

Company	Platoon	Losses	Kills	LER (kills:losses)
A	1	4	0	0:4
	2	2	4	4:2
	3	4	0	0:4
	4	3	1	1:3
	HQ	1	0	0:1
	total	14	5	5:14

Table 7: A Company's LER.

During the execution of this scenario, A Company lost most of their vehicles before detecting the opposing forces vehicles. They therefore had poor LERs.

Finally, Table 8 lists the ammunition expended percentage for A Company.

CO	PLT	Round Type	Start #	# Fired	End #	% Remaining
A	1 (M1A1)	Tank rnd 1	112	0	112	100.00
		Tank rnd 2	48	0	48	100.00
		AP round 2	4000	0	4000	100.00
		Lt armor 1	6000	0	6000	100.00
	2 (M1A1)	Tank rnd 1	112	1	111	99.11
		Tank rnd 2	48	5	43	89.58
		AP round 2	4000	0	4000	100.00
		Lt armor 1	6000	0	6000	100.00
	3 (M2 IFV)	missile 19	48	0	48	100.00
		AP round 2	2000	0	2000	100.00
		Lt armor 1	2000	0	2000	100.00
	4 (M2 IFV)	missile 19	48	1	47	97.92
		AP round 2	2000	0	2000	100.00
		Lt armor 1	2000	0	2000	100.00
	HQ (M2 IFV)	missile 19	24	0	24	100.00
		AP round 2	1000	0	1000	100.00
		Lt armor 1	1000	0	1000	100.00
	Co	Tank rnd 1	224	1	223	99.55
		Tank rnd 2	96	5	91	94.79
		missile 19	120	1	119	99.17
		AP round 2	13000	0	1300	100.00
					0	
		Lt armor 1	17000	0	1700	100.00
					0	

Table 8: A Company's Ammunition Expenditure.

Once again, the rationale that explained the high percentage of ammunition remaining for the DIS simulation run is the same here.

Overall, A Company performed poorly, having only 22 % of their vehicles remaining and poor LER. Sustainment for their remaining vehicles is good. Based on this performance, A Company's results are similar to the DIS simulation run and consequently have the same recommendations as previously discussed in section C. 1. b.

3. Janus vs. Janus in HLA with an Analysis Federate

The exercise for Janus vs. Janus in HLA with an Analysis Federate was run using the same set up as the first HLA run, but with the Analysis Federate included. Figure 13 shows how the exercise was set up.

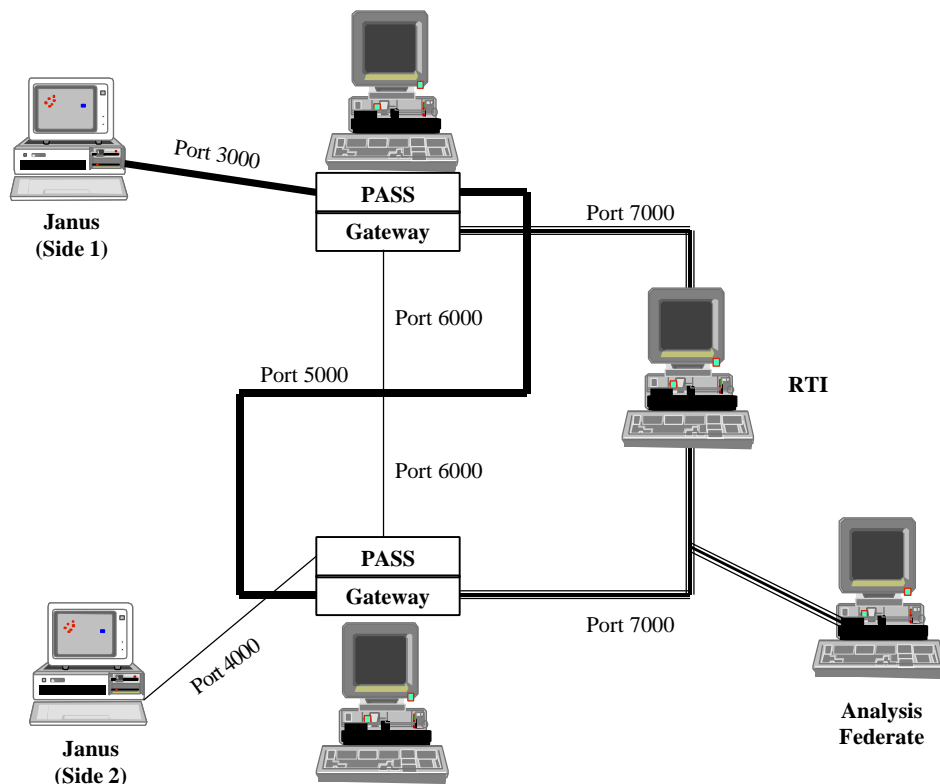


Figure 13: Janus vs. Janus in HLA with an Analysis Federate.

As a result of both PASS and Gateway running on one computer, the number of computers required was reduced from the original requirement for eight to six.

a. Data Collection

In order for the Analysis Federate to collect data, it first had to register with the RTI. During this process, the Analysis Federate provided a GUI allowing the analyst the opportunity to subscribe to the required objects and interactions. By point-and-click only the required items were selected. In addition to these items, the Analysis Federate automatically subscribes to enough basic information to replicate the exercise, such as movement locations and vehicle damage assessments. Once satisfied, the analyst uses the Analysis Federate to join the federation and sends the subscription requests and waits for the exercise to begin.

Once the exercise begins, the Analysis Federate receives the information it subscribed to and collects the subscription information. This information is stored in the Analysis Federate's database. For each simulation run, the Analysis Federate reserves 6 MB of memory for the data. If the database requires additional memory, the Analysis Federate will continue to reserve additional 6 MB blocks of memory until the exercise is complete or no more available memory exists.

The Analysis Federate allows the analyst the opportunity to observe the exercise in one of two modes: active or historical. The active mode shows the exercise as it unfolds. The game clock updates continuously as information is collected and processed. Since the Analysis Federate continually retrieves the information it needs for reports, the reports present the latest information. In the historical mode, only

information present from the time the historical mode is begun is available. This mode is typically used when the exercise is complete.

To answer the specific analysis questions for this study, the historical mode was used. The complete battle lasted approximately 35 minutes, during which the database for the entire battle never exceeded the 6 MB initially reserved for it. Even though collecting most of the subscriptions, the database did not contain information on 6 of the vehicles. For some reason, the Analysis Federate only received the subscriptions on entities that moved, fired, or were fired upon. All of the vehicles, however, did appear on both Janus models. The 6 vehicles missed did not move, fire, or get fired upon. Therefore, although the missing entities and events is a result of HLA, the calculations performed in the Analysis Federate will be incorrect for the respective units.

Gathering data to answer the analysis questions only required interacting with the Analysis Federate's operational view GUI. No coding or external programs were required to extract the data. Instead, several tools were available to accomplish this by using the mouse: establishing focus sets, selecting entities, adjusting the time period, selecting reports on units or fire missions, and zooming in to the areas of interest to name a few. Once a report is refined through the use of these tools, the report can be exported to an ASCII file to import into a spreadsheet for further calculations.

Before extracting any information, focus sets were established and the time period was set to include the entire battle. Focus sets are collections of entities with common attributes. For this study, a focus set was established for each platoon, company headquarters, company, and artillery battery. Each one was color coded for easy

identification at any point throughout the battle. Figure 14 shows how one operational view appeared after the focus sets were established.

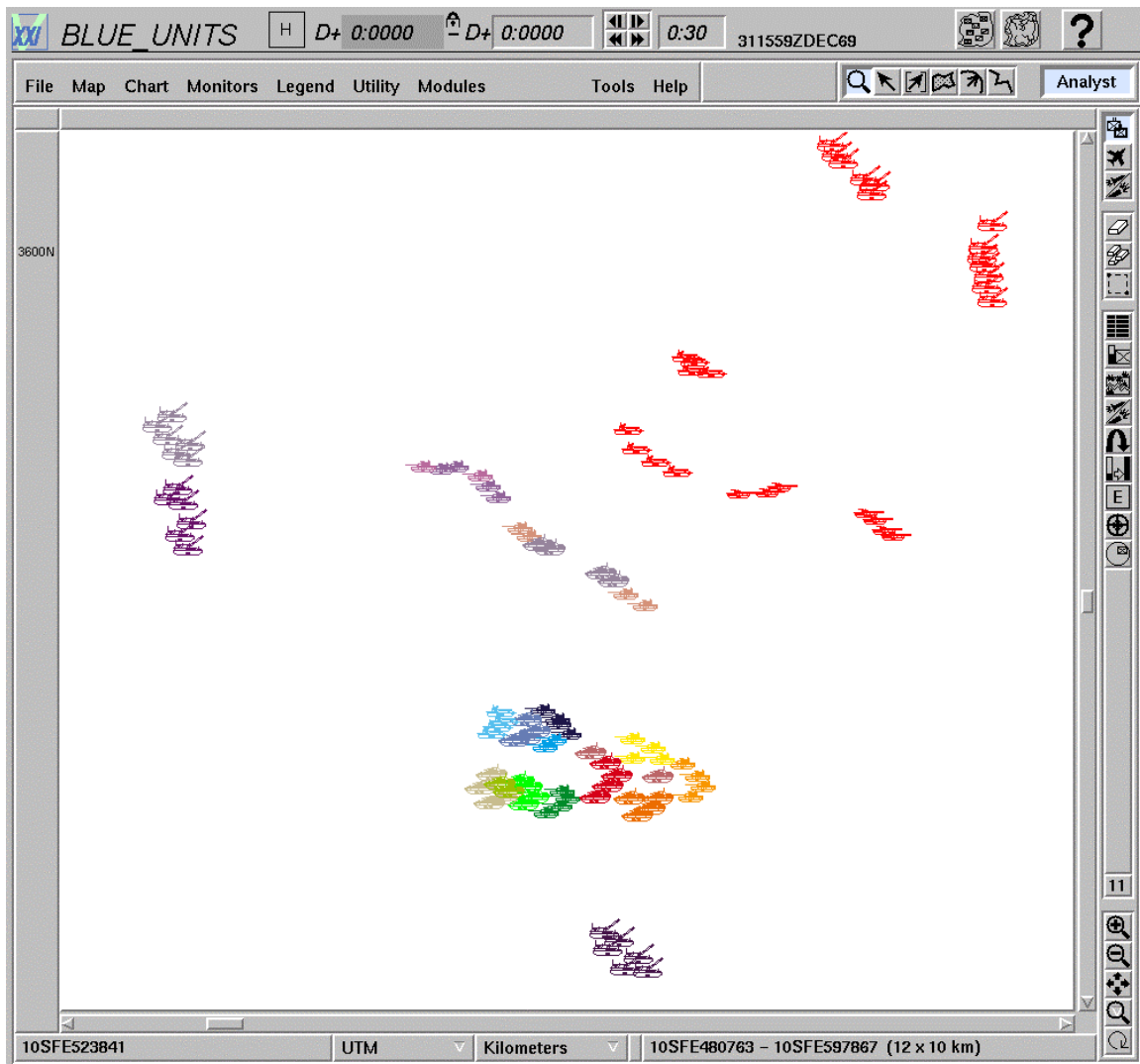
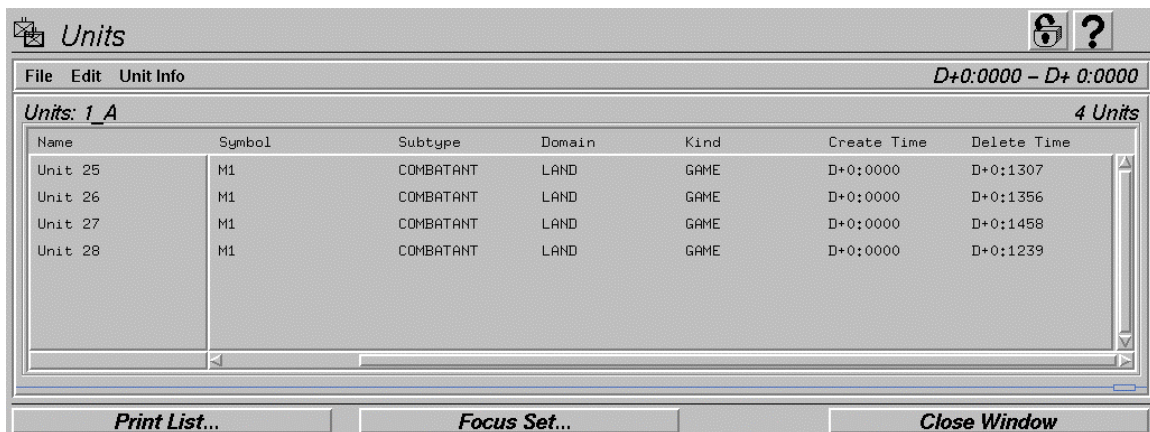


Figure 14: Operational View with Focus Sets.

By selecting a particular focus set, reports containing only information on those specific units can be viewed. Additionally the game time in the upper portion of Figure 14 was initialized at 0 hours 00 minutes and 00 seconds and ended at 1 hour, 20 minutes, and 00 seconds.

Three of the four analysis questions were answered with the information provided by the Analysis Federate. The three were the strength, LER, and ammunition expenditure questions. The only question that could not be answered was the fuel consumption question. As with DIS, data on fuel consumption was not available from PASS.

The strength of each unit required data on the number of starting and ending entities. To get this using the Analysis Federate, first the operational selection button was clicked. This operational selection window contained the list of focus sets shown in Figure 15. By selecting the focus set of interest and clicking the *LIST* button, the report for that appeared. Figure 15 shows an example of this window.



Name	Symbol	Subtype	Domain	Kind	Create Time	Delete Time
Unit 25	M1	COMBATANT	LAND	GAME	D+0:0000	D+0:1307
Unit 26	M1	COMBATANT	LAND	GAME	D+0:0000	D+0:1356
Unit 27	M1	COMBATANT	LAND	GAME	D+0:0000	D+0:1458
Unit 28	M1	COMBATANT	LAND	GAME	D+0:0000	D+0:1239

Figure 15: Primary List for a Focus Set.

If a time other than 0 is listed in the delete time column, the entity is destroyed. Since all entities are listed for that particular unit, the starting and ending numbers can be attained from this one report. This report was then exported to Excel for further calculations and processing.

The LER analysis question required the same information from the strength question and also who destroyed the dead entities. No reports are currently

present in the Analysis Federate for a killer/victim scoreboard. Therefore additional information is required to determine who destroyed whom. There is a report available, however, that lists each entity that fired and their target. For indirect fire, no target was given. For direct fire engagements, the target was given. The list includes all rounds fired whether they were a hit or a miss. To determine if a target was hit, the target's time of deletion from the status report above is compared with the time of the rounds being fired. For those targets that were hit, the entity firing will have the target entity listed as their target, and the killer/victim can thereby be determined.

The report that lists all rounds fired can be attained by selecting the fire missions icon. Then select all the units in the operational view and click on reports. The report appears in a separate window as described previously. Exporting the data from the Analysis Federate and importing it into Excel was done in the same manner described previously in this section.

The ammunition expenditure analysis question requires the number of rounds fired by each entity and type. This report was already exported to find out who killed whom. No further reports are required to extract the required information.

b. Results

Once the exported reports from the Analysis Federate are imported into Excel, calculations and processing were performed in a similar manner as described in the previous section. The results shown in this section pertain only to A Company. However, the same techniques were used to determine the results for the other units.

The status reports exported from the Analysis Federate did not contain the specific entity numbers. This information was not important in figuring out the unit

strength, but was important for the LER and ammunition expenditure. However, the entity numbers were retrieved from the unit status reports in the Analysis Federate and entered into the spreadsheet.

Tables 9 through 11 show the results for A Company's strength, LER, and ammunition expenditure for the HLA simulation run with the Analysis Federate.

Company	Platoon	Start #	Killed #	End #	% Remaining
A	1	4	4	0	0.00
	2	4	4	0	0.00
	3	4	1	3	75.00
	4	4	3	1	25.00
	HQ	2	0	2	100.00
	total	18	12	6	33.33

Table 9: A Company's Strength Remaining.

Company	Platoon	Losses	Kills	LER (kills:losses)
A	1	4	2	2:4
	2	4	3	3:4
	3	1	0	0:1
	4	3	1	1:3
	HQ	0	0	0:0
	total	12	6	6:12

Table 10: A Company's LER.

CO	PLT	Round Type	Start #	# Fired	End #	% Remaining
	1 (M1A1)	Tank rnd 1	112	2	110	98.21
		Tank rnd 2	48	1	47	97.92
		AP round 2	4000		4000	100.00
		Lt armor 1	6000		6000	100.00
	2 (M1A1)	Tank rnd 1	112	2	110	98.21
		Tank rnd 2	48	1	47	97.92
		AP round 2	4000		4000	100.00
		Lt armor 1	6000		6000	100.00
	3 (M2 IFV)	missile 19	48		48	100.00
		AP round 2	2000		2000	100.00
		Lt armor 1	2000		2000	100.00

4 (M2 IFV)	missile 19	48	8	40	83.33
	AP round 2	2000		2000	100.00
	Lt armor 1	2000		2000	100.00
HQ (M2 IFV)	missile 19	24	1	23	95.83
	AP round 2	1000		1000	100.00
	Lt armor 1	1000		1000	100.00
Co	Tank rnd 1	224	4	220	98.21
	Tank rnd 2	96	2	94	97.92
	missile 19	120	9	111	92.50
	AP round 2	13000	0	13000	100.00
	Lt armor 1	17000	0	17000	100.00

Table 11: A Company's Ammunition Expenditure.

A Company once again has a low overall performance. They lost 67% of their vehicles and killed 6 enemy while losing 12. Ammunition for the remaining 33 % appears to be high, but a force of only 33 % is combat ineffective. Therefore, as with the other two simulation runs, A Company needs to reevaluate the decisions that were made and see if they can improve their performance.

D. DISCUSSION

The results of conducting the three exercises provide a variety of strengths and weaknesses for each technique. The overall results also point out that each one could be improved to increase the timeliness and requirements for each.

Memory, network, and protocol requirements limited analysis in DIS. The data logger file for the approximate thirty-minute battle was 23 MB in binary form. To extract any useful information, the file had to be converted to ASCII, which was 93 MB large. For longer or more active simulations, the memory requirement will only increase placing stress on the system.

In both of the HLA approaches, memory and network requirements did not restrict the exercise execution or the analysis. For HLA without an Analysis Federate,

each Janus stored the required information to be able to process its post processor reports. No additional memory was required. The HLA with an Analysis Federate set aside 6 MB of memory prior to collecting any information. If additional memory would have been required it would have continued to reserve 6 MB blocks. For this exercise, the Analysis Federate never had to do this since the database never exceeded 6 MB.

In order to process the data and perform calculations, both DIS and HLA without an Analysis Federate had to wait until the exercise was complete. To use either the post processor reports or the data logger file, no more data could be collected. Only information collected prior to opening either file could be used in the analysis. In HLA with an Analysis Federate, however, real-time analysis can occur. The database for the Analysis Federate is stored on an internal server and the Analyst Workstation extracts the required information when requested by the analyst. Meanwhile, the Analysis Federate is continuing to update the database with data from the exercise.

All of the analysis approaches were able to answer three of the four questions. None of them were able to answer the fuel consumption question due to lack of information. This information is not included in the DIS PDU protocol or the post processor reports. However, the HLA exercise with the Analysis Federate was conducted using a short-term fix that did not fully implement the RTI. The Gateway software converted the DIS PDUs into HLA protocol and visa versa. If this had been a fully operational HLA exercise, Janus would have been HLA compliant and therefore had its own Simulation Object Model (SOM), such as the one developed by MAJ Larry Larimer. [13] The Attribute/Parameter Table of the SOM included a *FuelOnHand* attribute for the

various platforms, so if Janus had been fully HLA compliant, even the fuel consumption analysis question could have been answered.

In HLA, the Federation Object Model (FOM) is determined and agreed upon by the federates prior to setting up the federation. In order to enter the federation, federates must agree to publish and subscribe to the objects and interactions outlined in the FOM. In other words, the SOM of the federate must meet the requirements of the FOM. If not, the federate can be excluded from the exercise or else allowed to participate with only limited functionality in relation to the FOM. Since the proposed Analysis Federate approach required the analyst to determine his requirements prior to exercise execution, the federate would determine which federation to join in order to fulfill its data requirements. Therefore, a fully compliant HLA Janus could have answered all of its analysis questions.

Ease of use varied amongst the three analysis approaches. The DIS approach required an external parsing language to extract the required information. Although PERL was ideal for doing this, writing a program for more complicated procedures could be excessively time-consuming. The post processor reports were useful and easy to acquire. However, the information provided in the reports were either at the entity level or aggregated for all of the entities involved in the exercise. Also, Janus' post processor is relatively old and difficult to use on the screen. By far, the easiest and most useful tool was the Analysis Federate. Units could be aggregated, entities selected, and reports viewed all by the use of a mouse. Operational views also could be saved for use later in after action reviews or further analysis.

The Analysis Federate also allowed for the additional functionality of real-time analysis. By running the Analysis Federate on a PC Solaris, operational views and reports could be saved in a file and opened in applications on another machine. With this ability, feedback could be provided to commanders during the exercise instead of having to wait until the exercise was complete. For example, an operational view of the battle at a critical point in the fighting could be exported from the Analysis Federate to a file in a predetermined directory. At the same time a Power Point presentation could be set up on another computer on the same network with a hyperlink to the operational view's file. Then, by clicking on the hyperlink, the operational view could be opened to analyze what happened.

Finally although the results of each of the different runs were similar, they did differ somewhat, bringing up the question of repeatability. Repeatability is important when conducting analysis, since an analyst may want to repeat an experiment to check the impact of modifying it in some manner. For example, the vehicles killed during all three runs varied despite all the models being initiated with the same random number seed. Since Janus uses one random number stream, a small change in the order of events requiring random numbers will produce different results. The different order of events from run to run can most likely be accounted for by the time latency issues in a distributed environment. Therefore, time latency issues could interfere with the accuracy of the overall results and repeatability of the exercise.

Overall, the three simulation runs and subsequent analysis demonstrated the techniques and approaches for each infrastructure. The resulting comparison between them shows HLA with the Analysis Federate is the easiest and most functional tool. It

provides a workstation that an analyst can learn to use in a short amount of time and still present quality results. It also provides the opportunity for real-time analysis. This is a big advantage over the other techniques since feedback can be provided to the commanders while the exercise is still executing. Despite still being in the developmental stage, the current product is very useful and easy to use.

IV. CONCLUSIONS AND RECOMMENDATIONS

As DoD continues to deal with reduced budgets and force structures, Modeling and Simulation become increasingly important. A key player in the field of M & S is the analyst, whose purpose is to model the real world in order to optimize or predict future outcomes. The tools available to conduct this analysis depend on the environment in which the analysis is being conducted.

This study highlighted three main analysis tools across two different distributed environments, DIS and HLA. Post processor reports found commonly in the individual models were available in both DIS and HLA. In DIS, another tool, the data logger, was used in conjunction with a programming language PERL. For HLA, however, two different approaches were examined: an HLA exercise without any additional analysis tools and an HLA exercise in which an Analysis Federate in the distributed environment served as the primary analysis tool.

This chapter provides conclusions and recommendations for moving from DIS to HLA in the area of analysis. It is not a question of whether DoD will move from DIS to HLA, but when. With this in mind, the conclusions of this study were based upon the results of conducting a practical exercise and recommendations are made to facilitate the transition and improve analysis. Finally, future work will be discussed in the areas of improving HLA analysis.

A. CONCLUSIONS

DIS analysis takes advantage of the fact that the entire simulation is broadcast over the network using PDUs. By passively listening to the network, a data logger can record all of the events and updates into a file. After the exercise is complete, an analyst can parse the file and extract the required data for his analysis. Using this approach, the

data requirements, and questions for that matter, do not need to be determined until the exercise is complete. However, real-time analysis is not possible and all analysis is limited to data contained in the PDUs.

HLA, as it currently stands, does not possess any analytical tools. Instead, each individual model must provide the required data in the post processor reports. Although various models' post processors may contain a large list of available reports, the list is still finite. Therefore the amount and type of data is restricted and may not be adequate for answering some analysis questions. Another potential issue occurs when models of one type interact with models of another type resulting in each model's post processor being unable to capture and process events and entities from the different models. If this is the case, the final reports will be incomplete and inaccurate.

A proposed way to fix this analysis shortcoming in HLA is the Analysis Federate, currently under development at TRAC-Monterey. The first working model, which was used for this study, provided functionality and several advantages and alternatives for the analyst in HLA. The Analysis Federate provided real-time analysis, ease of use, and interoperability, not only with other combat models, but also with other multimedia programs.

With the move to HLA, the analysis methodology also must change. An analyst no longer has the flexibility to wait to develop analysis questions and data requirements. These must be developed prior to the exercise execution in order to subscribe to the proper required objects and interactions. An analyst is restricted to the subscription items once the initial registration is made. The analyst could update the subscription later

during the simulation run, but new subscription data is only available from the time of subscription until the present.

B. RECOMMENDATIONS

The overall recommendations from this study are twofold. First of all, incorporate the Analysis Federate into all HLA federations requiring analysis. The Analysis Federate developed by TRAC-Monterey provides the added functionality of interoperability within any federation. The Analysis Federate allows analyst to capture required data to answer specific analysis questions. Operational views and reports can be exported during execution to allow for real-time and future analysis. The second recommendation is to incorporate the study question object tree methodology to approaching analysis. In this process, requirements are determined prior to the exercise execution. This causes the analyst to be more proactive in conducting his analysis.

C. FUTURE WORK

As DoD continues to progress towards HLA, several issues deserve consideration and study. The first deals with the time latency issue. With models interacting through the RTI, time delays occur caused by network dispersion and processing times. The degree to which this takes place and effects the results of a simulation exercise is unknown. Next, as the Analysis Federate develops further, missing data, such as entities and artillery firings, need to be researched. Finally, although the Analysis Federate processes reports that can be exported into multimedia software packages, standardized reports could be developed to increase the functionality of the analysis tool. This would reduce the amount of processing and calculations that would otherwise be necessary external to the Analysis Federate.

In summary, the Analysis Federate fills an analysis void currently in HLA. By implementing it with the study question methodology, an analyst will be more effective and be able to provide real-time feedback.

APPENDIX A. PROTOCOL DATA UNIT (PDU) FORMAT

Distributed Interactive Simulation Constructive System (DISCS) interacts with other models in DIS through the use of three PDUs: Entity State, Fire, and Detonation PDUs. The Entity State PDU is used to provide information on the current status of an entity. It includes data such as velocity, location, and appearance. The Fire PDU is sent any time a round is fired. The data unit covers the firing and target entity, munition type, muzzle velocity, and range to the target. The Detonation PDU is sent when a round impacts a target or the ground or when a round explodes in the air and includes most of the same information as the Fire PDU and additionally the detonation result.

If Janus is interacting with other Janus models in DIS, DISCS also sends out event PDUs. These PDUs are used to send information that the Janus post processors need to complete their reports.

These PDUs are sent out almost at a constant rate anytime an entity's attribute changes. For entities whose attributes are not changing, a "heartbeat" PDU is sent out every 4 to 5 seconds. Tolerance levels are built into the code to reduce the number of PDUs being transferred yet still allowing accurate "pictures" of the battle to be captured. By capturing all of these PDUs, the entire simulation could be redisplayed.

Each PDU begins with the header data.

1. Protocol Version: This field shall specify the version of protocol used in a PDU.
2. Exercise Identification: Exercise Identification shall be unique to each exercise being conducted simultaneously on the same communications medium.
3. Type: This field shall indicate the type of PDU that follows.
4. Time Stamp: This field shall specify the time which the data in the PDU is valid. This field shall be represented by a timestamp
5. Length: This field shall specify the length of the PDU in octets. [14]

The information contained in the header identifies the exact exercise in which the PDU is involved. It also includes the game time of a particular event, such as firing an artillery round.

The body of the PDU follows the header data. The body contains the rest of the information required for one model to interpret an entity or event that occurred in another model. The PDUs use DIS enumerations to standardize different events and entities. For example, these enumerations specify exactly what type of round was fired and from what type of platform or weapon system.

Here are some examples of PDUs.

ENTITY STATE PDU * from (131.120.57.51)**

PDU HEADER:

Protocol Version: 4

Exercise Identification: 9
 Type: 2
 Time Stamp: 442625387 (6.3584 sec)
 Length: 96
 Entity ID: Site 58, Host 45, Entity 42
 Force: Opposing (Red) = 2
 Entity Type: Kind 2 Dom 2 Cntry 225 Cat 2 Scat 13 Spec 2 extra 0
 Alternate Entity Type: Kind 2 Dom2 Cntry 225 Cat2 Scat13 Spec 2 extra0
 Velocity = 0.0000, 0.0000, 0.0000.
 Location = -2685767.5994, -4414798.9580, 3726793.2616.
 Orientation = -1.0300r(-59.0121d), 0.6487r(37.1665d), -2.4372r(-139.6415d).
 Appearance (in Hex): 40008038
 Appearance_PaintScheme_Uniform
 Appearance_DamageDestroyed
 Appearance_SmokePlume
 Appearance_Flaming
 Appearance_PlatformLand_NotConcealed
 Appearance_Platform_Defilade_Exposed
 Marking: '027_2_WM'
 Capabilities (in Hex): 00000000
 Dead Reckon Parameters:
 Algorithm: 1 DRAlgo_Static
 5 Articulated Parameters:
 #1: Change: 256, ID: 0, Type: 4107, Value: 6.2832(360.0000deg).
 (PPrimaryTurretNumber)
 #2: Change: 257, ID: 1, Type: 4429, Value: 0.0000(0.0000deg).
 (PPrimaryGunNumber1)
 #3: Change: 10752, ID: 0, Type: 4107, Value: 0
 (Janus_Number_Elements)
 #4: Change: 10752, ID: 0, Type: 4107, Value: 0
 (Janus_Defilade_Status)
 #5: Change: 10752, ID: 0, Type: 4107, Value: 0 (Janus_Flight_Mode)

FIRE PDU ### from (131.120.57.51)

PDU HEADER:

Protocol Version: 4
 Exercise Identification: 9
 Type: 2
 Time Stamp: 335055505 (3.5214 sec)
 Length: 96
 Firing Entity ID: Site 57, Host 39, Entity 13
 Target Entity ID: Site 58, Host 45, Entity 42
 Event ID: Site 57, Host 39, Event 34
 Muntion Entity ID: Site 0, Host 0, Entity 0
 Muntion Entity Type: Kind 2 Dom2 Cntry225 Cat 2 Scat 13 Spec 2 extra 0

Fuze: 1100 FuzeMunition_ContactInstant
Warhead: 1400 WarheadMunition_HighExpAntiTank
Firing Location = -2683259.3629, -4416818.0808, 3726220.9693.
Muzzle Velocity = -1193.5254, 941.2942, 253.2661.
Range = 3301.8159.

DETONATION PDU +++ from (131.120.57.51)

PDU HEADER:

Protocol Version:	4
Exercise Identification:	9
Type:	2
Time Stamp:	335625387 (5.1429 sec)
Length:	96

Firing Entity ID: Site 57, Host 39, Entity 13
Target Entity ID: Site 58, Host 45, Entity 42
Event ID: Site 57, Host 39, Event 34
Munition Entity ID: Site 0, Host 0, Entity 0
Munition Entity Type: Kind 2 Dom2 Cntry225 Cat 2 Scat 13 Spec 2 extra 0
Impact Velocity = -1192.9290, 920.0544, 253.1452.
Impact Location = -2685767.5994, -4414798.9580, 3726793.2616.
Fuze: 1100 FuzeMunition_ContactInstant
Warhead: 1400 WarheadMunition_HighExpAntiTank
Detonation Result: 1 DetResult_EntityImpact

Through the use of PDUs like those shown above, DISCS is able to interact in a DIS environment. More technical information on PDUs is available on-line at <http://www.pitch.se/fmv/dis-items/Pduindex.htm>.

APPENDIX B. JANUS DATABASE

Janus provides a robust database for entering parameters. Figure 16 shows the structure of the database. Following the figure is a list of parameter topic areas for each of the combat systems subtopics. These parameters are used to create a scenario specific to the user's needs. When building a new scenario, the user can create a new scenario completely from scratch by entering data for each of the parameters below or by copying and changing the appropriate ones from a previously built scenario.

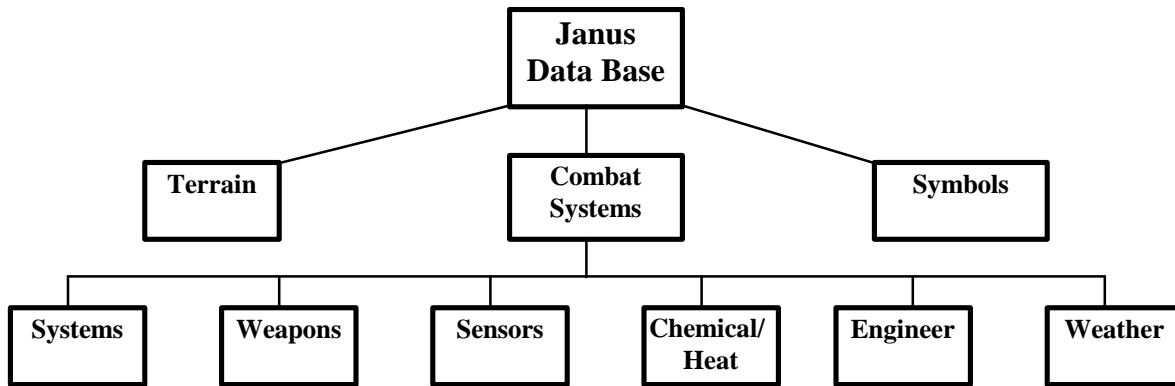


Figure 16: Janus Database Hierarchical Diagram. [15]

Systems:

General	Mine Vulnerability	Kill Category
Functionality	POL	Vulnerability to indirect fire
Volume/Weight	Weapons/Ordinance	Detection
Weapons Selection		

Weapons:

Characteristics	MOPP effects	PH data sets
Round Guidance	PH/PK data sets by Weapon/target	PK data sets

Sensors:

Optical/Thermal	Flyer fuselage/Rotor	Jammer/Radar
Contrast/Temperature	Rotor track radius	Characteristics
On-board seekers	Rotor acquisition times	Jammer effectiveness
Range dependent	Fuselage probability track	Probability of detect data
Capability footprints	Fuselage Radar x-section	BCIS Characteristics
Jammer Characteristics		

Chemical/Heat:

Chemical rounds	Chemical susceptibility	Heat stress
-----------------	-------------------------	-------------

Engineer:

Barrier delays	Mine detection/duds	Non-arty smoke
Activation kills	VEES	Grenades
Smoke Pots	Large area generators	Minefields
Dispersing	Clearing	

Weather: Weather Characteristics

Terrain: The terrain editor also allows you to adjust the terrain as needed. Buildings, fences, roads, trees, urban/city areas, generic (general purpose) strings, generic (general purpose) areas, and rivers can be added. In other words, the terrain can be built to meet a wide variety of specific needs.

Symbols: Janus uses standard military symbols to display different entities and occurrences on the graphics package.

The following table shows the size limitations on the database.

	Master Database:	Scenario Database:
Systems	400	100
indirect fire systems	100	30
weapon types (direct fire systems)	400	70
Weapons per system	15	15
PH data sets	4000	600
PK data sets	4000	600
Flyer types	64	
Weather/location types	16	1
Sensor types	45	
Mine types	10	10
Air defense Radar types	20	20
Artillery vulnerability categories	28	
Artillery projectile types/system	12	
Precision guided missile types/side	2	
Aerosol chemical type per side	1	
Artillery round types	9	

Table 12: Statistics for Overall Database. [15]

Janus stores the data in a complex database with several interactions between entries. Therefore by changing one entry, another parameter or characteristic may also be effected. However, the database parameterizes a lot of information which in turn allows the analyst to build the scenario exactly as needed.

APPENDIX C. JANUS POST PROCESSOR REPORTS

There are a large number of reports available in the Janus Post-Processor. The reports are not created until the exercise is complete or the Janus model execution has ended. The analyst then has the opportunity to select which of the reports he wants. Once selected, the analyst can view the reports on the computer screen or send them to a printer. Data in the reports include “bean counts” and processed data presented in table formats. Following is a list of the available reports and the information included in each. [8]

Execution Time Parameters: This report is automatically published when a post processor report is made. It contains the original parameters entered prior to the run as well as data on the performance parameters, workstation selection, engineer barrier assignments, force performance data, and enemy definition.

Artillery Impacts report: This report lists each artillery volley in chronological order. It includes the time of impact, firing unit data, aimpoint, number of rounds, type of projectile, precision guided data, target’s unit number, and field artillery scatterable minefield (FASCAM) data.

Artillery Summary Report: This report displays the total ammunition expenditure by volleys and rounds by ammunition type for all sides firing artillery missions.

Direct Fire Report: This report presents detailed information on each direct fire engagement for the entire battle. The report has two parts: detailed engagement and a direct fire ammunition expenditure report.

- a. Detailed engagement report: This report includes game time, firer data to include speed, target data to include speed, shooter-target status, number of elements in firing unit, single shot kill probability, range from shooter to target, weapon fired, time target remained suppressed if the shot missed.

- b. Direct fire ammunition expenditure report: This part of the report gives the number of rounds fired by weapon type and side.

Coroner’s Report: This report provides a detailed account of each kill. It also is broken up into two parts: Coroner’s report and Systems killed by time interval.

- a. Coroner’s report: This report depicts all sides separately. It includes game time, kill type, victim information to include location, killer information to include location, range in kilometers from the killer to the victim, and type of round killing the target.

- b. Systems killed by time interval: This report breaks down the simulation run into 10-minute intervals. It includes the system type and number killed for all systems by side.

Killer/Victim Scoreboard: This report produces a series of other reports broken down into three parts: direct fire, indirect fire, and miscellaneous.

- a. Direct fire: This report displays number of direct fire losses by system type. This is broken down into side 1 vs. side 2, side 1 vs. side 3, side 2 vs. side 1, etc.
- b. Indirect fire: This report is the same except it covers indirect fire losses.
- c. Miscellaneous: This report combines the data from the direct and indirect fire reports and adds minefield losses. This is also in the side vs. side format.

Minefield Summary Report: This report is also in two parts: minefield summary and minefield encounters.

- a. Minefield summary report: This report provides information about each minefield. It includes the time emplaced, side emplacing it, mine type, density code, total number of mines, location, dimension of the minefield, and orientation angle.
- b. Minefield encounters: This report provides data on units that encounter minefields. It includes time encountered, encountering unit, breach mode, buttoned up or not, entrance point, exit point, and minefield number.

Detection Report: This report provides detection information in three different reports: individual detections, detection summary, and detections by system for each side.

- a. Individual detections: This report shows detections by side 1 of side 2, side 1 of side 3, side 2 of side 1, etc. It includes time of detection, detector information to include detecting sensor type and status (moving, defilade, flying, etc.), detected information with status, and range between detector and detectee.
- b. Detection summary: This report provides information on detections by sensor types and side. It includes sensor type, class, number of detections, minimum range, maximum range, and average range.
- c. Detections by system: This report summarizes the number of detections by weapon system. It includes information on system type, number of detections both near and far, and sensor data.

Heat and Chemical Casualties: This report provides information on losses to heat and chemicals. It includes the time of chemical/heat event, event type, type of system, location. Loss, threshold of vulnerability, and actual amount of chemical or heat present.

Temperature and Workload Profiles: This report furnishes information about the temperature and workload experienced by each unit during the course of the battle. This report currently is not printed out.

Game Analysis: This report prints out five additional reports: Force Loss Analysis, System Exchange Ratio, Contribution by System, Detections Scoreboard, and Engagement Range Analysis. These reports contain more processed than “bean count” data. They have potential measures of performance and effectiveness. These reports

allow you to specify different cases for analyzing different weapons or sides against each other. This gives the analysts the flexibility to build the reports as they see fit.

a. Force Loss Analysis: This report includes the systems counted in the calculations, initial number of blue and red systems, total number of blue and red losses, initial force ratio, loss exchange ratio, and force exchange ratio.

Initial Force Ratio: initial Red / initial Blue “smaller is better” for blue
Loss Exchange Ratio: red losses / blue losses “bigger is better” for blue
Force Exchange Ratio: % red losses / % blue losses “bigger is better” for blue
(% red losses = red losses / initial red, same for % blue losses)

b. System Exchange Ratio: This report provides the system exchange ratio (SER) for the selected systems. It is broken down by blue and red forces. It also includes the systems counted in the calculations.

Blue Forces (by system): number of enemy systems killed by friendly systems / total number of friendly systems. “Bigger is better” for blue.

Red Forces (by system): number of kills by red on blue / total number of red systems killed by blue forces. “Smaller is better” for blue. When three sides were included, the report did not account for kills by side 2 on side 1. It only included the kills by side 3 on side 1.

c. System Contribution Report: This report provides information about how many of the selected enemy systems were killed by the selected friendly systems. It provides the percentage of counted red coalition systems killed by blue coalition system type and visa versa.

d. Detections Scoreboard: This report is essentially a consolidated detection report. It includes for each side on side the total number of detections by system against system of blue and red.

e. Engagement Range Analysis: This report furnishes information on both direct firings and kills for selected systems against selected enemy systems. It includes game time, kill type, victim information, killer information, range, round type, total number of kills, and average range.

APPENDIX D. HLA GATEWAY MODIFICATIONS

HLA Gateway Version 2.3 required some slight modifications to three files in order for it to work on the operating systems on the computers at TRAC-Monterey. The first file was the Gateway Configuration File. The following list shows the changes that were required to this file.

1. Site = DIS site ID.
2. Host = the machine ID number.
3. Exercise = Exercise ID of this DIS simulation.
4. Ip_address = Network interface controller.
5. Udp_ports = UDP port numbers for DIS.
6. Utm_coords = Terrain database used in the exercise.

Each of the listings were specific to the exercise and network address of the computers that were used.

The next file that was modified was the RTI Configuration File. Since the RTI was written to work on IRIX-6.2o operating system and the only available operating system was IRIX-6.4o, the architecture and path had to be changed to the following lines.

```
setenv RTI_ARCH IRIX-6.2o
setenv XPM_HOME ${RTI_HOME}/lang/C++/demo/Jager/sys/${RTI_ARCH}
```

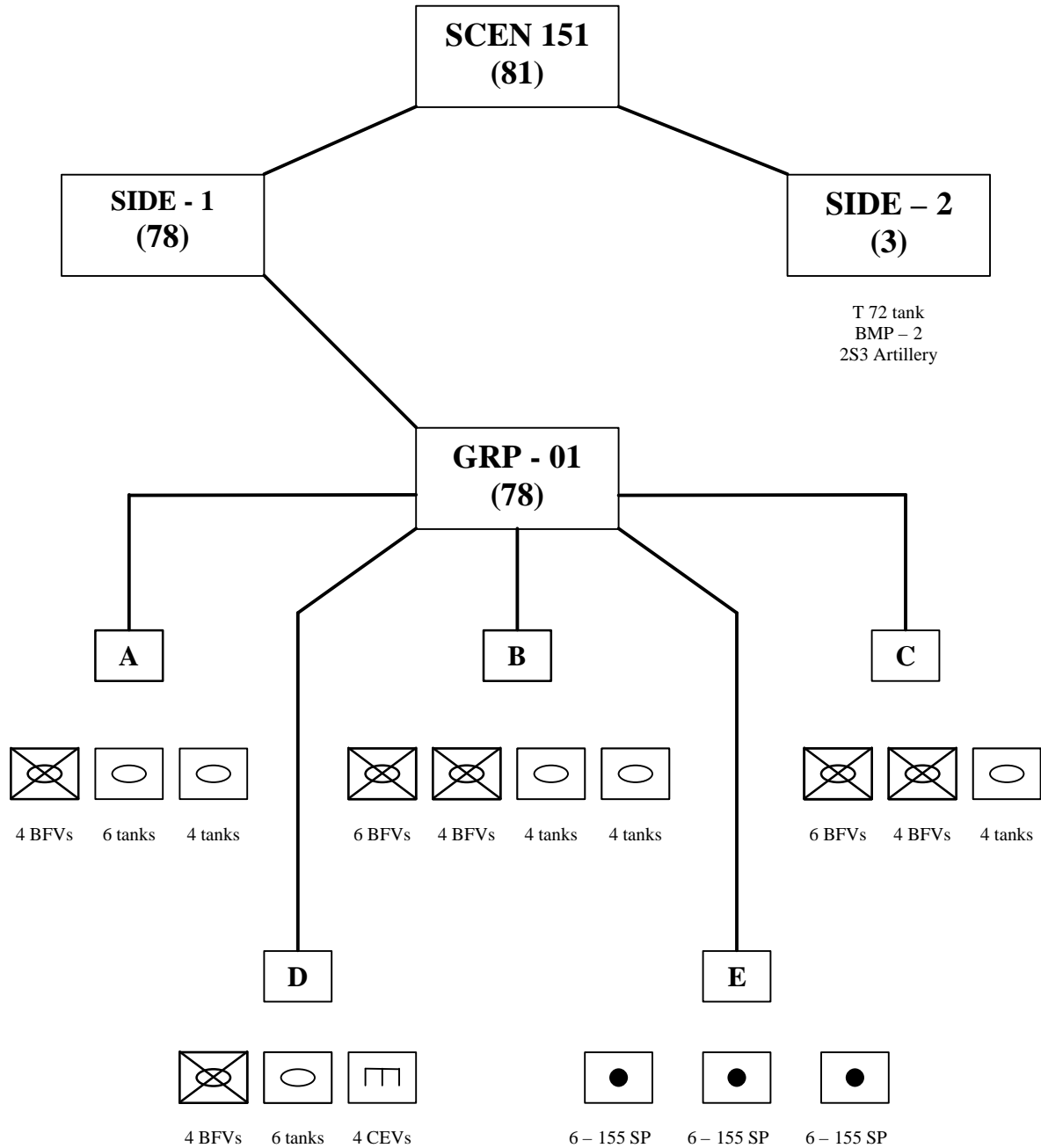
In this way the limited operating system version requirement was bypassed.

Finally, the RTI.rid file had to be modified. This file is the main configuration file for the RTI. The lines that required updating were BEST_EFFORT_PORT, RTI_EXEC_HOST, and RTI_EXEC_PORT. Each of these lines were specific to the exercise and network available for the exercise.

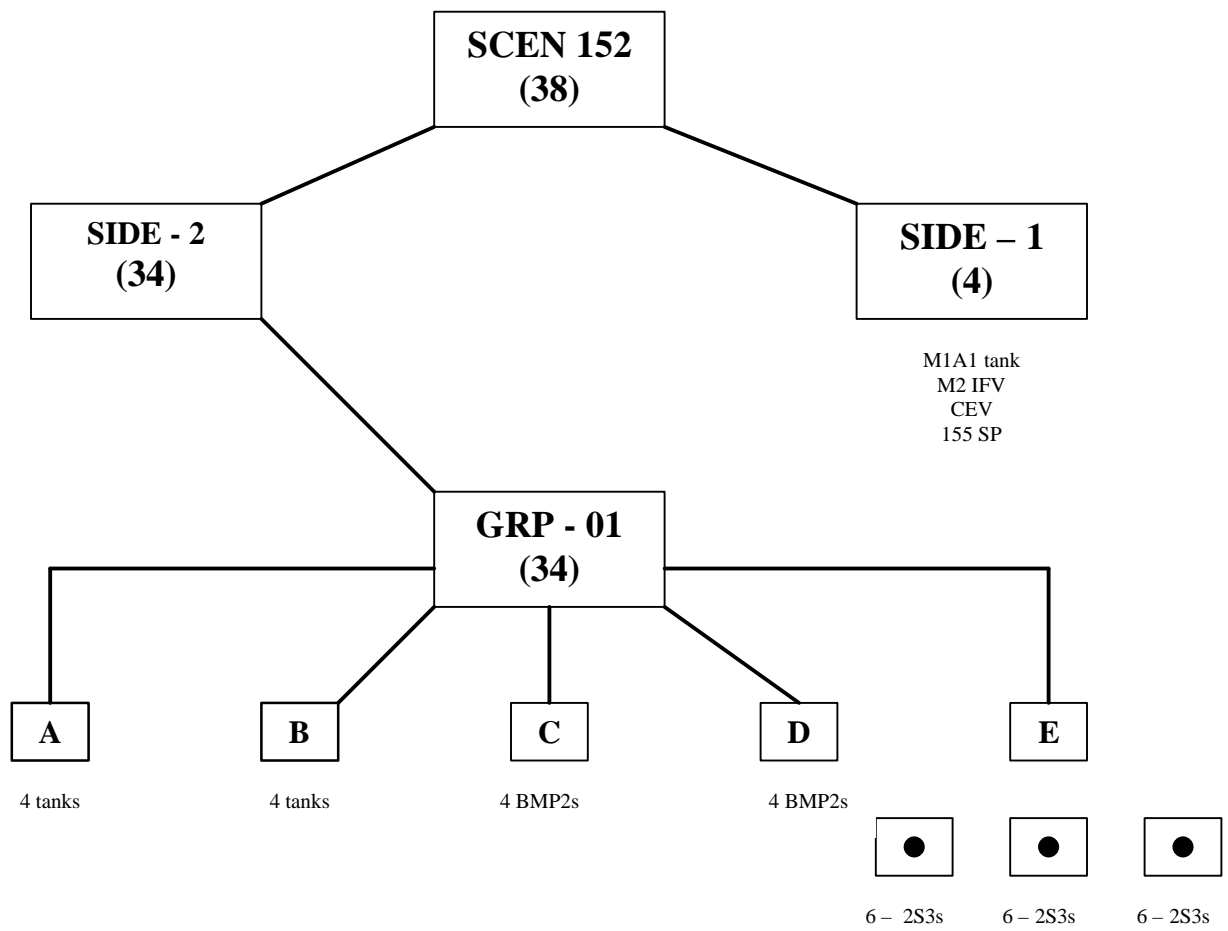
Each of the file discussed above were changed for each of the HLA Gateways and respective RTIs.

APPENDIX E. SCENARIO SIDE ONE AND TWO BREAKDOWN

SCENARIO 151: Side 1



SCENARIO 152: Side 2



APPENDIX F. PERL CODE FOR KILLER/VICTIM DATA

This is the PERL code to parse the data logger file to extract the killer/victim data.

```
#!/usr/bin/perl -w

# Steven D. Knight
# Thesis: Parsing the data logger using PERL
# 5 May 98

# This program reads in the data logger file (ASCII) and extracts
# what vehicle killed what vehicle and when. The output is a
# table with this information separated by a tab. The output can
# be saved to a file and imported into excel for further calculations.

# Booleans for the type of PDU currently parsing
$detPDU = 0;
$entityStatePDU = 0;
%detonations = ();
$entity = "";

while(<>) {

# Detonation PDU check and look:
if (/DETONATION PDU/) {
    $detPDU = 1;
}

# record time stamp
if ($detPDU && /Time Stamp:\s*[0-9]*\s*(\W.*)/) {
    $timeStamp = $1;
}

# record firer entity
if ($detPDU && /Firing Entity ID:\s*(\W.*)/) {
    $firer = $1;
}

# record target entity
if ($detPDU && /Target Entity ID:\s*(\W.*)/) {
    $target = $1;
}
```

```

# Determine round result
if ($detPDU && /Detonation Result:\s*[0-9]*\s*(\W.*)/) {
    $detResult = $1;

    # Determine if the round was a direct hit.
    # If so, record data. Otherwise, set values back to null.
    if ($detResult eq 'DetResult_EntityImpact') {
        $detonations{$target} = {'Firer'=>$firer,'Time'=>$timeStamp,'Status'=>'alive'};
        $detPDU = 0;
        $timeStamp = "";
        $firer = "";
        $target = "";
        $detResult = "";
    } else {
        $detPDU = 0;
        $timeStamp = "";
        $firer = "";
        $target = "";
        $detResult = "";
    }
}

```

Entity State PDU check and look:

```

if (/ENTITY STATE PDU/) {
    $entityStatePDU = 1;
}

if ($entityStatePDU && /Entity ID:\s*(\W.*)/) {
    $entity = $1;
}

```

Check to see if the entity is dead

```

if ($entityStatePDU) {

    # If this line is found, then we have gone past where DamageDestroyed
    # would be, so stop looking. The impacted round must not have
    # destroyed the entity (target).
    if (/Dead Reckon Parameters:/) {
        $entityStatePDU = 0;
        $entity = "";
    }
}

```

Check the targets that were hit to see if they were destroyed.

```

foreach $key (keys %detonations) {

```



```

        if ($key eq $entity) {
            if (/Appearance_DamageDestroyed/) {
                ${$detonations{$key}}{'Status'} = 'dead';
                $entityStatePDU = 0;
            }
        }
    }

}

}

}

# Printing Output

foreach $key (keys %detonations) {
    if (${ $detonations{$key}}{'Status'} eq 'dead') {
        print
        "$key\t${$detonations{$key}}{'Firer'}\t${$detonations{$key}}{'Time'}\t${$detonations
        {$key}}{'Status'}\n";
    }
}

```


APPENDIX G. PERL CODE FOR ROUND TYPE DATA

This PERL code was used to extract the number of rounds fired by round type and entity from the data logger file.

```
#!/usr/bin/perl -w

# Steven D. Knight
# Thesis: Parsing the data logger using PERL
# 5 May 98

# This program reads in the data logger file (ASCII) and extracts
# what rounds were fired and by whom. The output is a table with
# the firing entity, type of round fired, and the number of round fired
# separated by a tab. The output can be saved to a file and imported
# into excel for further calculations.

# Booleans for the type of PDU currently parsing
$firePDU = 0;
%rounds = ();
$roundRepeat = 0;
$fireTime = "";
$fireEntity = "";
$fireRound = "";

while(<>) {

# Fire PDU check and look:
if (/FIRE PDU/) {
    $firePDU = 1;
}

# record firing time
if ($firePDU && /Time Stamp:\s*[0-9]*\s*(\W.*)/) {
    $fireTime = $1;
}

# record firing entity
if ($firePDU && /Firing Entity ID:\s*(\W.*)/) {
    $fireEntity = $1;
}
```

```

# record type of round fired
if ($firePDU && /Muntion Entity Type:\s*(\W.*)/) {
    $fireRound = $1;
}

# When the input has reached the line with Muzzle Velocity, all of the
# required information has been recorded. At that point, record all
# all of the information into a hash table. Also, check to see if
# the entity has fired that type of round before and increment the
# round count accordingly.
if ($firePDU && /Muzzle Velocity/) {
    foreach $key (keys %rounds) {
        if ($key eq $fireEntity && ${$rounds{$key}}{'Round Type'} eq $fireRound) {
            $roundRepeat = 1;
        }
    }

    if ($roundRepeat) {
        ${$rounds{$fireEntity}}{'Count'} += 1;
        $roundRepeat = 0;
        $firePDU = 0;
        $fireTime = "";
        $fireEntity = "";
        $fireRound = "";
    } else {
        $rounds{$fireEntity} = {'Time'=>$fireTime,'Round Type'=>$fireRound,
'Count'=>1};
        $firePDU = 0;
        $fireTime = "";
        $fireEntity = "";
        $fireRound = "";
    }

}

}

# printing output into a table
foreach $key (keys %rounds) {
    print "$key\t${$rounds{$key}}{'Round Type'}\t${$rounds{$key}}{'Count'}\n";
}

```

APPENDIX H. GLOSSARY OF ACRONYMS

BFV	Bradley Fighting Vehicle
CEV	Combat Engineer Vehicle
DCA	Data Collection and Analysis
DIS	Distributed Interactive Simulation
DISCS	Distributed Interactive Simulation Constructive System
DMSO	Defense Modeling and Simulation Office
DoD	Department of Defense
FED	Federation Execution Data
FOM	Federation Object Model
GUI	Graphical User Interface
HLA	High Level Architecture
IST	Institute for Simulation and Training
JAWS	Janus Analysis Workstation
JCATS	Joint Conflict and Tactical Simulation
JTLS	Joint Theater Level Simulation
LAN	Local Area Network
LER	Loss Exchange Ratio
M & S	Modeling and Simulation
ModSAF	Modular Semi-automated Forces
MOE	Measure of Effectiveness
OMT	Object Model Templates
PASS	Protocol Data Unit Adapter Software System
PDU	Protocol Data Unit
RPR	Real-Time Platform
RTI	Run-Time Infrastructure
SIMNET	Simulation Networking
SOM	Simulation Object Model
STOW	Synthetic Theater of War
TCP/IP	Transmission Control Protocol/Internet Protocol
TRAC	Training and Doctrine Command Analysis Center
TRADOC	Training and Doctrine Command
UDP/IP	User Datagram Protocol/Internet Protocol
WAN	Wide Area Network

LIST OF REFERENCES

1. Piplani, Lalit K. COL, Mercer, Joseph G. LTC, and Roop, Richard O. LTC, Defense Systems Management College, Fort Belvoir, VA, System Acquisition Manager's Guide for the use of Models and Simulations, September 1994.
2. Distributed Interactive Simulation (DIS) Master Plan, Headquarters Department of the Army, September 1994.
3. Perla, Peter P., The Art of Wargaming, Annapolis, MD: Naval Institute Press, 1990.
4. DODD 5000.59 DoD Modeling and Simulation (M&S) Management, January 4, 1994, (<http://web7.whs.osd.mil/text/d500059p.txt>).
5. Pearman, Gerald M. MAJ, Naval Postgraduate School, Monterey, CA, "Comparison Study of Janus and Jlink," June 1997.
6. Jackson, Leroy A. MAJ and Wood, J. Ralph LTC, TRADOC Analysis Center – Monterey, "Exploiting the High Level Architecture for Analysis in Advanced Distributed Simulation," 1997.
7. Murphy Jr., William S. MAJ, TRADOC Analysis Center – Monterey, briefing to 9th U.S./French Seminar on Operations Research and Simulation, "HLA Federate for Data Collection and Analysis," April 1998.
8. Software User's Manual, Version 6.3, UNIX Model, Simulation, Training, & Instrumentation Command, Orlando, FL.
9. TRAC-Monterey page explaining DISCS, (<http://www.trac.nps.navy.mil/jlink/index.html>).
10. Pate, Maria C. MAJ, Command and General Staff College, Fort Leavenworth, Kansas, and Roussos, Glen G. MAJ, TRADOC Analysis Center - Monterey, "JLINK - A Distributed Interactive Janus," Winter 1996.
11. The paper about analysis study questions by MAJ William S. Murphy
12. HLA Gateway Release 2.1 User Guide, Defense Modeling, Simulation, and Tactical Technology Information Analysis Center, Orlando, FL, 31 October 1997.
13. "Clinton plans to bring some Bosnia troops home," Army Times, 2 March 1998.
14. Larimer, Larry R. MAJ, Naval Postgraduate School, Monterey, CA, "Building an Object Model of a Legacy Simulation", June 1997.

15. Distributed Interactive Simulation data dictionary homepage explaining PDU standards, (<http://www.pitch.se/fmv/dis-items/Pduindex.htm>).
16. Janus Database User's Manual, Version 6.3, UNIX Model, Simulation, Training, & Instrumentation Command, Orlando, Florida.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
8725 John J. Kingman Road, Suite 0944
Fort Belvoir, VA 22060-6218
2. Dudley Knox Library.....2
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
3. Professor Arnold H. Buss, Code OR/Bu.....2
Operations Research Department
Naval Postgraduate School
Monterey, CA 93943-5101
4. Director.....2
U.S. Army TRADOC Analysis Center – Monterey
Monterey, CA 93943-5101
5. Major William S. Murphy.....2
U.S. Army TRADOC Analysis Center – Monterey
Monterey, CA 93943-5101
6. Captain Steven D. Knight.....3
811 Woodview Drive
Massillon, OH 44646
7. Tapestry Solutions, Inc.....1
5675 Ruffin Road, Suite 305
San Diego, CA 92123
8. Director.....1
U.S. Army TRADOC Analysis Center – Leavenworth
255 Sedgwick Avenue
Fort Leavenworth, KS 66027